fedora

10

Linux Administration, Networking, and Security

To Aliena and Jesse

Fedora 10 Linux Administration, Networking, and Security

Richard Petersen

Surfing Turtle Press

Alameda, CA

Please send inquires to: editor@surfingturtlepress.com

Library of Congress Control Number: 2009901225

ISBN 0-9820998-3-5

ISBN-13 978-0-9820998-3-4

Trademark Acknowledgements

UNIX is a trademark of The Open Group

Microsoft and MS-DOS are registered trademarks of Microsoft Corporation

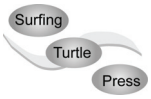
IBM and PC are registered trademarks of the International Business Machines Corporation

Red Hat and Fedora are trademarks of Red Hat, Inc.

 and fedora are trademarks of Red Hat, Inc

See **www.fedoraproject.org/wiki/Logo/** for more information

 is the Solar background image for Fedora 10



is a trademark of Surfing Turtle Press

Preface

This book is designed as an administration and security reference. Administration tools are covered as well as the underlying configuration files and system implementations. The emphasis is on what administrators will need to know to perform key administration and security tasks. Topics covered include user management, time server settings, start up configuration, software management, kernel configuration, SELinux, and file system management. Server tools are covered as well as the underlying configuration files and system implementations. Topics covered include network connections, IP network administration, software management, Upstart service management, runlevels, and the Network Time Protocol. Key servers are examined, including Web, FTP, CUPS printing, NFS, and Samba Windows shares. Network support servers and applications covered include the Squid proxy server, the Domain Name System (BIND) server, DHCP, and IPtables firewalls.

The book is organized into six parts: system administration, security, and file system and device management, network services, shared resources, and network support.

Part 1 focuses on administrative tasks such as managing users, managing software with PackageKit, customizing the kernel, and setting up virtual systems.

Part 2 keys in on security tasks beginning with authorizations using PolicyKit. GPG encryption support with seahorse as well as the structure of public/private key encryption is covered. File and directory permissions, along with access controls are examined. SELinux tools and the format and command structure of SELinux configurations are discussed. SSH encryptions and Kerberos authentication are also examined. The security section ends with a detailed examination of IPtables firewalls and the system-config-firewall tool.

Part 3 deals with file systems and devices. File systems formats are discussed in detail along with mount and encryption operations. LVM and Linux RAID are covered. For devices, both HAL and udev are examined in detail. Backup applications for your file systems are then discussed.

Part 4 examines Internet servers as well as how all services are managed by Upstart using runlevels. Configuration and implementation of the Postfix mail server, the vsftpd FTP server, the Apache Web server, as well as news and database servers are covered in detail.

Part 5 deals with servers that provide shared resources on a local network or the Internet. Services examined include the Cups printing server, NFS Linux network file server, and Samba Windows file and printing server, and the GFS distributed file system.

Part 6 covers servers that provide network support: configuring network connections, the Squid proxy server, the Bind Domain Name System (DNS) server, DHCP servers, and IPv6 network addressing and monitoring.

Overview

Part 2: Security

Part 3: Devices and File Systems

Part 4: Network Services

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Contents

Part 2
Security

Part 4
Network Services

Part 5
Shared Resources

25. Network File Systems and Network Information Service: NFS and NIS 579

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Part 1: System Administration

1. Fedora 10 Introduction

Linux Overview

Fedora Linux

Fedora Documentation

Fedora 10 changes

Standard Fedora features

Fedora Desktop

Fedora Software Repositories

Linux is an fast, stable, and open source operating system for PC computers and workstations that features professional-level Internet services, extensive development tools, fully functional graphical user interfaces (GUIs), and a massive number of applications ranging from office suites to multimedia applications. Linux was developed in the early 1990s by Linus Torvalds, along with other programmers around the world. As an operating system, Linux performs many of the same functions as UNIX, Macintosh, Windows, and Windows NT. However, Linux is distinguished by its power and flexibility, along with being freely available.

Technically, Linux consists of the operating system program, referred to as the kernel, which is the part originally developed by Linus Torvalds. But it has always been distributed with a massive number of software applications, ranging from network servers and security programs to office applications and development tools. Linux has evolved as part of the open source software movement, in which independent programmers joined together to provide free quality software to any user. Linux has become the premier platform for open source software, much of it developed by the Free Software Foundation's GNU project. Many of these applications are bundled as part of standard Linux distributions. For Fedora Linux, most open source applications are available its online repositories, the Fedora software repository. Most of the Internet server, system administration, security, and desktop (GNOME and KDE) applications are incorporated into the Fedora repository, using software packages that are Fedora compliant. You should always check the Fedora repository (System | Administration | Add/Remove Software) first for the software you want. Additional software, including proprietary graphics drivers, can be obtained directly from the Fedora compliant RPM Fusion software repository. Once configured, software on the RPM Fusion repository can be directly accessed using Fedora's Add/Remove Software tool.

Linux is free, including the network servers, security applications, and desktop interfaces. Linux is distributed freely under a GNU General Public License as specified by the Free Software Foundation, making it available to anyone who wants to use it. GNU (the acronym stands for "GNU's Not Unix") is a project initiated and managed by the Free Software Foundation to provide free software to users, programmers, and developers. Linux is copyrighted, not public domain. However, a GNU public license has much the same effect as the software's being in the public domain. The GNU general public license is designed to ensure Linux remains free and, at the same time, standardized. Linux is technically the operating system kernel—the core operations—and only one official Linux kernel exists.

Fedora provides an extensive set of administration, security, and network configuration utilities for use on the GNOME desktop. These provide a way to perform most administrative and configuration tasks easily. These include common administrative tasks like user management and time settings, as well as more complex operations like firewall configuration and security intrusion detection. For difficult operations like SELinux management, Fedora provides SELinux desktop administration utilities. Fedora also provides desktop administrative tools for many Internet servers including the vsftpd FTP server and the Apache Web servers.

Fedora Linux

The Fedora release is maintained and developed by an Open Source project called the Fedora Project. The release consists entirely of open source software. Development is carried out using contributions from Linux developers, allowing them free rein to promote enhancements and new features. The project is designed to work much like other open source projects, with releases keeping pace with the course of rapid online development. The Fedora project features detailed

documentation of certain topics like Installation and desktop user guides at **http://doc.fedoraproject.org** (see Table 1-1).

The Fedora versions of Linux are entirely free. You can download the most current version, including betas, from **http://fedoraproject.org** or **http://download.fedoraproject.org**. The **http://download.fedoraproject.org** address will link to the best available mirror for you. You can update Fedora using the Update System (PackageKit) to access the Fedora repository. Access is automatically configured during installation.

Web Site	Name
http://fedoraproject.org	Fedora Resources
http://download.fedoraproject.org	Fedora repository, mirror link
http://mirrors.fedoraproject.org	Fedora mirrors list
http://download.fedora.redhat.com	Fedora repository site
http://fedoraproject.org/wiki/Tours/Fedora10	Fedora 10 tour, with images and videos
http://fedoraproject.org/wiki/Tools/yum	YUM tools
http://docs.fedoraproject.org/install-guide/f10/en_US/	Fedora Installation guide
www.fedoraproject.org/wiki/Overview	Fedora Project Overview
www.fedoraproject.org/wiki/FAQ	Fedora FAQ
http://docs.fedoraproject.org	Documentation and support tutorials for Fedora releases.
http://fedoraforum.org	End-user discussion support forum, endorsed by the Fedora Project. Includes FAQs and news links.
http://fedorafaq.org	Unofficial FAQ with quick help topics such as enabling MP3 and 3-D graphic card support.
http://fedoranews.org	Collects the latest news and developments on Fedora.
www.linux-foundation.org	The Linux Foundation, Official Linux development.
www.kernel.org	Latest Linux kernels.
www.redhat.com	The Red Hat Web site
www.redhat.com/magazine	Red Hat Magazine with specialized articles on latest developments.
www.centos.org	Community Enterprise Operating System, CENTOS (Red Hat based)

Table 1-1: Fedora sites

Fedora Documentation

Documentation for Fedora can be found at **http:/docs.fedoraproject.org**, with specialize topics provided at **http://fedoraproject.org/wiki/Docs** (see Table 1-1). The Fedora installation guide provides a detailed description of all your install procedures. The Fedora desktop users guide covers basic desktop operations like logging in, using office applications, and accessing the Web. Several dedicated Fedora support sites are available that provide helpful information, including **www.fedoraforum.org**, **www.fedoraproject.org**, and **www.fedoranews.org**.

The **www.fedoraforum.org** site is a Fedora Project–sponsored forum for end-user support. Here you can post questions and check responses for common problems. One of the more useful sites is the unofficial Fedora FAQ at **www.fedorafaq.org**. Here you will find solutions to the more common problems like graphic card issues and enabling MP3 support.

Your Firefox Browser will already be configured with tabs for accessing popular documentation and support sites. These include the Fedora Project home page, the Fedora Weekly News, community support from Fedora forums as well as communication page, and the Red Hat Magazine.

Fedora maintains detailed specialized documentation, like information on understanding how udev in implemented or how SELinux is configured. For much of the documentation you will have to rely on installed documentation in **/usr/share/doc** or the Man and info pages, as well as the context help button for different applications running on your desktop. Web sites for software like those for GNOME, KDE, and OpenOffice.org will provide extensive applicable documentation. For installation, you can use the Fedora Installation Guide at **http://docs.fedoraproject.org/install-guide**.

Red Hat also maintains an extensive library of documentation for Red Hat Enterprise Linux, much of which is applicable to Fedora. From the Red Hat home page, you can click to its support page, and then on its Documentation page. The documentation page provides a pop-up menu from which to select products. Documentation covers topics like virtualization, the Global File System (GFS), Logical Volume Management (LVM), and the Installation Guide. Tip, HOW-TO, and FAQ documents are also provided. All the Red Hat documentation is freely available under the GNU General Public License. Before installing Red Hat Enterprise Linux on your system, you may want to check the online Installation guide. The Red Hat Magazine provides information on the latest features.

Fedora 10 changes

The new release of Fedora features key updates to critical applications as well as new tools replacing former ones. NetworkManager will now configure all connections, including wired Ethernet and wireless 3G. The Printer configuration tool has a new interface, allowing for a more intuitive management. The former third party Fedora repositories, like Livna and Freshrpms, have been combined into one: RPM Fusion.

Installing Fedora continues to be simplified. You can install from a standard DVD or from Live CD/DVDs. A core set of applications are installed, and you add to them as you wish. Following installation, added software can be taken either from the disk or from online repositories, depending on which is more up to date. You will find that the software on your disk becomes quickly obsolete. Most software will be retrieved from the online Fedora repository. Install screens

have been reduced to just a few screens, moving quickly through default partitioning, network detection, and time settings, to the package selection.

The Fedora distribution of Linux is available online at numerous FTP sites. Fedora maintains its own FTP site at **http://download.fedora.redhat.com** along with a mirrors listing at **http://mirrors.fedoraproject.org**, where you can download the entire current release of Fedora Linux, as well as updates and additional software. The Web page **http://download.fedoraproject.org** links to an available mirror for you. Install DVDs can be downloaded from mirrors. The following information is derived for the official Fedora 10 Release notes. Consult these notes for detailed information about all new changes. The Fedora release notes are located on the all the Fedora spins as an HTML file on the top directory. You can also fine the release notes at:

```
http://docs.fedoraproject.org/release-notes/f10/
```

For a tour of Fedora 10 with images and video, check the following site.

```
www.fedoraproject.org/wiki/Tours/Fedora10
```

Desktop Changes

- The major third party Fedora compliant repositories have been integrated into one repository, RPM Fusion (**http://rpmfusion.org**). The repositories integrated include Livna (**http://rpm.livna.org**) Freshrpms (**www.freshrpms.net**), and Dribble (**http://dribble.org.uk**). Software packages are no longer available on these sites (Freshrpms still provides older packages). RPM Fusion provides both free and non-free collections. Simply download and install the RPM Fusion YUM configuration package from the RPM Fusion Web site, to automatically configure YUM and PackageKit to access all the third party repository software. There are separate configuration packages for free and non-free collections.
- With the NetworkManager, you can automatically select wireless connections. NetworkManager now supports mobile broadband 3G connections. NetworkManager can also configure IP, PPP, and DSL connections.
- Fedora 10 supports wireless connection sharing, allowing other systems to use a Fedora wireless system as a wireless network node, setting up an ad hoc local wireless network.

```
http://fedoraproject.org/wiki/Features/ConnectionSharing
```

- Improved interface for the printing tool, system-config-printer. New job queue and printer configuration window, as well as support for both personal and system defaults.

```
http://fedoraproject.org/wiki/Features/BetterPrinting
```

- New theme and background called Solar.
- PulseAudio sound server now uses timer-based audio scheduling instead of the interrupt-driven methods. Provides cleaner audio with power savings. The Pulse Audio server applet provides direct access to all Pulse Audio applications.

```
http://fedoraproject.org/wiki/Features/GlitchFreeAudio
```

- Gstreamer codec packages can be made Gstreamer detectable. When a new codec is required, its package can be installed directly by PackageKit (YUM), if available. Codeina has been replaced by this PackageKit support.
- Totem supports backends, no longer requiring recompilation to support additional playback capability. In particular the Totem Xine backend provides Xine support on Totem (**totem-xine** package). Use the **totem-backend** command to change backends.
- Updated version of bluetooth.
- Empathy instant messenger (GNOME 2.24) with telepathy plugin framework.
- GNOME Display Manager (GDM) is updated with PolicyKit control. Still no configuration support (to configure you have to manually edit **/etc/gdm/custom.conf**).

http://live.gnome.org/GDM/2.22/Configuration

- Codeina codec installation helper has been replaced by direct PackageKit support for Gstreamer codec detection and installation.
- For KDE, users can use KPackageKit, the KDE version of PackageKit, to manage software.

http://fedoraproject.org/wiki/Games

- KDE 4.1 desktop with file manager (dolphin), theme (oxygen), and desktop/panel (Plasma), with hardware integration (Solid) and multimedia support (Phonon).
- To better allow access by users to system level applications, **/sbin** and **/usr/sbin** have now been added to all user PATHs, allow direct use of configuration features for applications like Network Manager by users.
- The LXDE desktop (Lightweight X11 Desktop Environment) is available for smaller systems like netbooks, OLPC, embedded systems, and older computers.
- Sugar desktop for OLPC computers can now be used on Fedora installs.
- Automatic detection of Infrared remote control devices (LIRC), now supported directly by an LIRC daemon. Configuration supported by **lirc**, **lirc-remotes**, and **gnome-lirc-properties (**System | Preferences | Hardware | Infrared Remote Control). Plugins provided for Totem, Rythmbox, and XMMS. Also PulseAudio support.

http://fedoraproject.org/wiki/Features/BetterLIRCSupport

- Additional Webcam drivers.

http://fedoraproject.org/wiki/Features/BetterWebcamSupport

Administration and System Changes

- The Red Hat Graphical Boot (RHGB) has been replaced by Plymouth, a much more streamlined, efficient, and faster graphical boot that does not require X server support. It relies on the kernel's Kernel Modesettings (KMS) feature that provides direct support for basic graphics. With the Direct Rendering Manager drivers Plymouth can make use of various graphical plugins. KMS support is currently provided for ATI graphics cards with Intel in development. Nvidia does not support KMS.

http://fedoraproject.org/wiki/Features/BetterStartup

- Input devices (mice and keyboards) are now managed as hotplugged devices by HAL. HAL provides support directly to the Xorg **evdev** driver. Input devices are no longer configured in **/etc/X11/xorg.conf**. All references to input devices are removed from this file. Input devices are now configured by HAL **.fdi** files (**10-x11-input.fdi**). If you are using a customized **xorg.conf** file, be sure to edit it to remove the input device entries.
- Firstaidkit automated recovery tool provides recovering operations for common problems. The tool uses pluggable components so that new recovery operations can be added easily. FirstAidKit is meant to work in Rescue mode from a Rescue CD/DVD disc or Live CD/DVD, first diagnosing the problem and then suggesting a plugin to run to fix it. The user can then choose to run the plugin, fixing the problem.

http://fedoraproject.org/wiki/Features/FirstAidKit

- Kernel Modesettings (KMS) provides faster boot, fast user switching, and seamless X server switching (available for ATI graphics cards).
- Virtualization now supports storage access outside of the Virtual disks. Virtual systems can now mount and access partitions, LVM volumes, and iSCSI storage.
- The sectool SELinux utility performs security audits on your system for intrusion detection, **sectool** and **sectool-gui** packages.
- RPM 4.6 is a major rewrite of the Red Hat Package Manager (RPM). Commands and interface remain the same. RPM 4.6 also supports a new compression method, LZMA (Lempel-Ziv-Markov chain-Algorithm), for faster download and space savings. Better lower level integration with YUM.

http://rpm.org/wiki/Releases/4.6.0

- Support for managing online account information and access. Online account information like that for Google can now be stored by GConf with GNOME Keyring, and on the online account service provided by **online.gnome.org**. Currently only the Online Desktop is supported (for Google and Twitter), but other applications may soon be supported.
- MySQL has been upgraded to version 5.0 with numerous changes.

http://dev.mysql.com/doc/refman/5.0/en/releasenotes-cs-5-0-67.html

- OpenJDK 6, open source JAVA, is now installed for the JAVA Runtime Environment service. The java-gcj-compat collection is still available and provides Java runtime environment compatibility.

Installation Issues

- You can install from a Live image running in RAM or on a USB drive
- For IDE drives, the device name **hd** has been deprecated, and is replaced by **sd** (except for PPC Fedora versions). IDE drives are no longer referred to as **/dev/hd***x*, but as **/dev/sd***x*.
- Most motherboard RAID controllers are supported by Fedora using the **dmraid** module. If your motherboard RAID is not supported, you can use Linux software RAID.

- If you computer has multiple network interfaces (NIC or PXE), it may not assign eth0 to the first network interface (see release notes).
- All hard disks, including SATA and IDE, are limited to 15 partitions per disk (SCSI device specifications), instead of 63 (IDE device specifications). If your hard disk has more than 15 partitions, you should migrate them to an LVM volume.
- During installation, NTFS support is implemented with **ntfs-3g**. NTFS partitions on your system will automatically be detected and mounted for you at subdirectories in the **/media** directory. The subdirectory names will use the partition labels or the name **disk**.

Upgrade Issues

- When upgrading be sure to backup your **/home** and **/etc** directories (also **/opt** and **/usr/local** for any customized packages).
- If your system references hard disk partitions by their device names instead of labels, you will have to first label those partitions. Partition device labels are required for upgrading. The kernel now handles device names hd*x* and sd*x* differently from previous releases. The install process sidesteps this confusion by identifying drives by their labels instead of their device names. This means that all drives now have to be labeled before installations. If upgrading from a previous version of Fedora, be sure your hard drive partitions are all labeled. If not, then label them before upgrading. Upgrading cannot proceed unless all disks are labeled.
- If you are using LVM partitions (the default for previous Fedora installations), your devices are already labeled.
- Use the following block ID command to check if your devices are already labeled. Each device should have a LABEL field.

```
/sbin/blkid
```

- If you add a label, be sure to also add the same label in the devices corresponding entry in the **/etc/fstab** file.
- Should you have to add a device label for a boot partition, or a root partition that includes the boot directory, be sure to add the label entry in the kernel line of the **/etc/boot/grub.conf** file. Usually your root or boot device will likely be already labeled. By default, previous installations will assign a / label to the root partition, or /boot to a boot partition.
- Be sure to test your changes by rebooting and making sure everything mounts and starts up correctly.

Fedora ISO images

Fedora disks are released as a set of spins that collect software for different purposes. Currently there are five spins available: The standard DVD image for desktops, workstations, and servers, the Live CD/DVD (GNOME Desktop), Fedora KDE Live CD, Fedora Developer Live CD, and Fedora Electronic Lab (FEL) Live CD. The regular Fedora DVD spin includes a collection of workstation and server software, but not the entire collection. Spins are created with Pungi, which

you can use to develop your own customized spins. You can find out more information about spins at: **https://fedoraproject.org/wiki/SIGs/Spins/10/**, which also describes Custom Spins.

Fedora Live CD: The Fedora GNOME desktop live CD, available in i386 and x86_64 versions.

Fedora live DVD: The Fedora install live DVD with more complete selection of software, available in i386 and x86_64 versions.

Fedora KDE live CD: The Fedora KDE desktop live CD, available in i386 and x86_64 versions.

Fedora Custom Spins

Custom spins are also available. Torrents for these can be downloaded from **http://spins.fedoraproject.org**,

Fedora Developer Spin: i686 only. Includes GNOME desktop with development applications like Eclipse IDE, along with debugging and profiling tools, as well as documentation.

Fedora FEL Spin: Fedora Electronics Lab spin.

Fedora Edu/Math Spin: Fedora Educational and Math spin.

Fedora BrOffice: Brazillian-Protugese spin with OpenOffice.org.

Fedora AOS Spin: JeOS spin for building system images for virtual appliances.

Fedora XFCE Desktop Spin: Features the light-weight XFCE desktop instead of GNOME or KDE.

In addition you can download the Everything spin from the Fedora Unity Project, as well as periodic re-spins of Fedora ISO images with updated software packages, **http://spins.fedoraunity.org/spins**.

Fedora Everything Spin: Fedora Unity Project complete set or Fedora software, downloaded through BitTorrent or Jigdo. Images are available in CD, DVD, DVD-DL, and BlueRay formats.

Fedora Re-Spins: Fedora Unity Project periodic re-spins of Fedora ISO images using the most recent updated packages.

Fedora specialized spins

In addition there are several specialized spins, still under development, with version currently available. They have their own locations. These include the Omega and Sugar Fedora spins.

Omega: Fedora Multimedia spins with complete set of multimedia codecs and applications from RPM Fusion (also includes the DVD codec from Livna). Currently at:

```
ftp://ftp.infradead.org/pub/spins/
```

Sugar Spin: OLPC desktop, currently at:

http://alt.fedoraproject.org/pub/alt/olpc/0.82/

Linux Kernel

Three kernel builds are available. One is the Native version used on most systems, one is a specialized kernel for PAE (32 bit over 4 GIGs of memory), and the other is a debugging kernel used in development. For each of these kernels there may be 32bit, 64bit, and ppc versions. Both the Native and PAE versions include support for Kdump, SMP (Hyper-threading), and Xen Para-virtualizations (separate kernels for these features are no longer required). The kernel spec file is named just **kernel.spec** instead of including the kernel version.

Multimedia

Though licensed multimedia formats like MP3 and DVD are still excluded. Open Source formats are all included, such as Vorbis, Ogg, Theora, Speex, and FLAC. Multimedia codecs with licensing issues can be directly downloaded with Packagekit, once you have configured YUM to use the RPM Fusion repository on your system (just install the **http://rpmfusion.org** configuration package for Fedora 10 which is available from that site). The RPM Fusion repository is not configured by default.

You can obtain full licensed codecs free or cheaply for GStreamer applications from Fluendo (**www.fluendo.com**).

X Window System

The X server sever is designed to automatically detect and configure your graphics hardware, significantly reducing the entries in the **/etc/X11/xorg.conf** file. Only the graphics driver information is placed in the **xorg.conf** file. Other devices including the monitor, keyboard, and mouse are configured automatically by HAL. The monitor is queried for correct resolution and frequency. You can change the resolution with the System | Preferences | Hardware | Screen Resolution tool. If your monitor or graphics card is incorrectly detected, you can use System | Administration | Display to choose the correct monitor or graphics card. Backup the **/etc/X11/xorg.conf** file first.

Standard Fedora features

Listed here are some of the features that are now standard for Fedora.

- A free and open source Flash player, **swfdec**, is now included with Fedora.
- Consolidated dictionary support for applications like Thunderbird, Firefox, GNOME, and OpenOffice.org.
- The Thunderbird mail client is included in this release.
- FreeIPA integrates auditing, identity, and policy management using a Web page interface. You can easily configure NTP, Kerberos, and DNS. Combines the Fedora directory server and Free RADIUS.
- Development support for the **ext4** file system, which is more scalable with better performance. Useful for large files.

- Service management is now handled by the Upstart init daemon, replacing the older System V init daemon. Upstart is event driven, and is much more flexible in handling services. Originally developed for Ubuntu, it is now standard for Fedora. System V Init structure is still emulated by Upstart which most services still use.
- Browsing Windows shares and remote printers with Samba is enabled from Gnome, through the Gnome virtual file manger. You can use system-config-firewall to enable Samba Windows browsing.
- Localized Common User Directories (xdg-user-dirs) automatically sets up Documents, Video, and Music directories in user home directories.
- A demo version of the GNOME Online Desktop provides a panel and links to online resources like Google mail and flickr (online-desktop).
- Removable devices like USB printers, digital cameras, and card readers are automatically detected. CD/DVD discs are treated as removable devices, automatically displayed and accessed when inserted. Input devices are now treated as removable devices.
- The IDE hard disk drivers are now supported by the Serial ATA drivers. All IDE devices are named with an **sd** prefix, not the **hd** prefix.
- A wide range of multimedia applications are included, such as a video player and a TV viewer, along with compatible support from various multimedia applications and libraries available from **http://rpmfusion.org**, such as DVD and DivX support.
- Information about hot plugged devices is provided to applications with the Hardware Abstraction Layer (HAL) from freedesktop.org. This allows applications like GNOME to easily display and manage removable devices.
- PackageKit is used to automatically update your Fedora system and all its installed applications, from the YUM Fedora online repositories.
- With the alternative compiz window manager, desktop AIGLX effects can be enabled. From the Administration | Preferences | Look And Feel | Desktop Effect entry, choose Enable Effects. You will see wobbling windows when moved, and rotating workspaces with the workspace switcher.

Fedora Desktop Look and Feel:

Fedora 10 features a desktop look and feel with a Fedora logo, as well as the new default Fedora solar screen background. The logo depicts an F encased in a blue circle. On the main panel you will now see the blue Fedora logo as the icon for the Applications menu. The default background for Fedora 10 is the Solar background, with variations for morning, day, evening, and night. The logo even has its own package, fedora-logos.

Figure 1-1: Fedora Logo

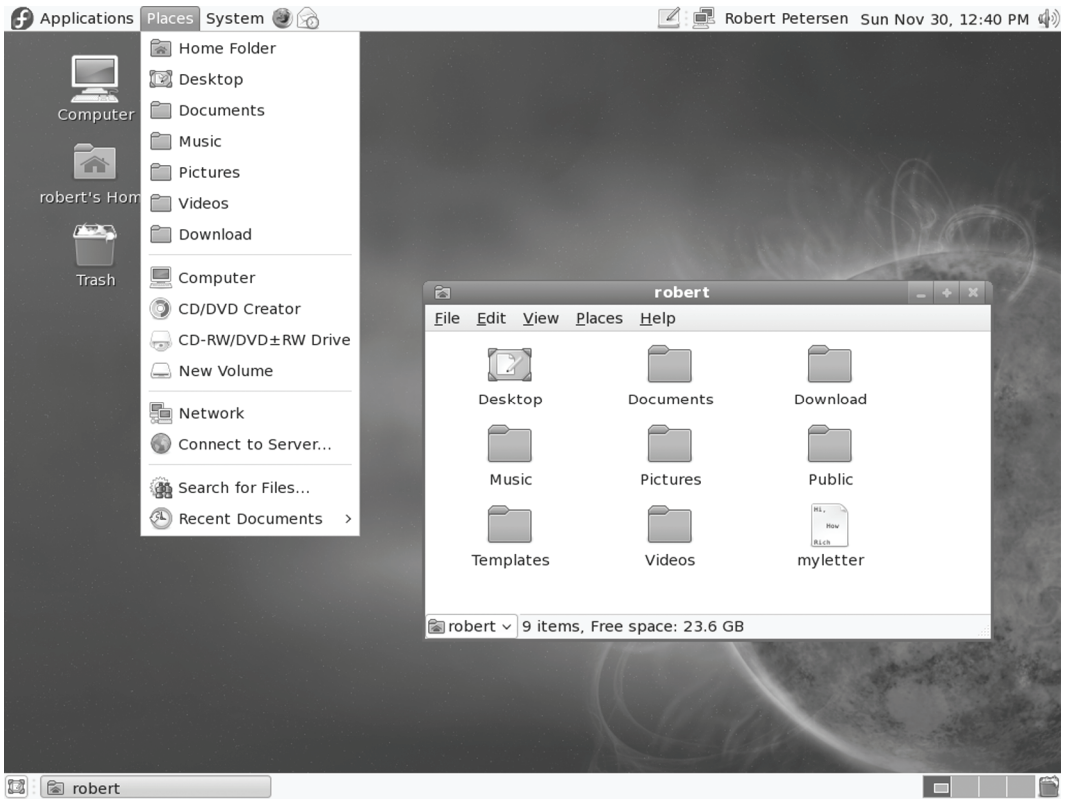


Figure 1-2: Fedora 10 desktop with Fedora logos

The logo is designed to represent three features of the Linux community and development: freedom, communication, and infinite possibilities - the **f** for freedom, which melds into the Infinity symbol, both encased in a speech bubble evoking communication (voice). Free and open software with infinite possibilities developed through global communication. The idea is to evoke the spirit and purpose of Linux development as one of infinite freedom given a voice. The logo incorporates the four basic ideals of Fedora: open, free, innovative, and forward looking (see Figure 1-1). See **http://fedoraproject.org/wiki/Logo** for more details. A description of the logo as shown on the site is shown here.

The default theme is Nodoka, a GNOME theme designed for Fedora. Buttons and windows are easier to use, and appear more pleasing to the eye. Of course numerous other themes are available from the Gnome theme manager (see Figure 1-2).

GDM Automatic Login

To implement an automatic login from the desktop, you have to edit the **/etc/gdm/custom.conf** file and add certain configuration setting. Create a **[daemon]** segment and enter the following TimedLogin options.

```
[daemon]
TimedLoginEnable=true
TimedLogin=myuser
TimedLoginDelay=10
```

First you enable the TimedLogin feature. Then specify the user you want to have automatically logged in (TimedLogin=). You also have to set up a TimedLoginDelay, usually about 10 seconds; otherwise you will always be logged in to that user, and not have a chance to login to another.

You can only edit this file as the root user. Use **su** command and use a line editor like **vi**, or use **sudo gedit /etc/gdm/custom.conf**, if sudo is enabled**.**

The GDM will show this user as automatic login. When logging into another user on the GDM, the automatic login user is displayed. Click Cancel to display the list of al users, from which you can choose to login.

Fedora Software Repositories

For Fedora, you can update to the latest software from the Fedora YUM repository using Update System (see Chapter 4). Your Update System tool is already configured to access the Fedora repositories.

The Fedora distribution provides a comprehensive selection of software ranging from office and multimedia applications to Internet servers and administration services (see Table 1-2). During installation, YUM is configured to access Fedora repositories.

Due to licensing restrictions, multimedia support for popular operations like MP3, DVD, and DivX is not included with Fedora distributions. A Fedora Project–associated site, RPM Fusion, **http://rpmfusion.org**, does provided support for these functions. Here you can download support for MP3, DVD, and DivX software, as well as for the proprietary Nvidia or ATI Linux graphics drivers. This site integrates support previously provided by Livna, Freshrpms, and Dribble (see Chapter 4).

For multimedia codecs, you can, instead of using RPM Fusion, download licensed MP3 and DVD GStreamer plug-ins from **www.fluendo.com**.

Note: Fedora does include the generic X.org Nvidia and ATI drivers, which will provide all the capabilities most people need. You do not have to use the Nvidia or ATI proprietary drivers.

RPM Fusion has Fedora-compliant YUM configuration files that you can download and install as an RPM package. You can then use Add/Remove Software (PackageKit) or the **yum** command to select, download, and install software from the repository directly.

URL	Internet Site
http://fedoraproject.org/get-fedora	Download page for the latest Fedora releases, includes direct, jigdo, and bittorrent downloads.
http://mirrors.fedoraproject.org	Page listing Fedora mirrors
http://download.fedoraproject.com	Link to best available mirror
http://download1.rpmfusion.org http://rpmfusion.org	Fedora applications not included with the distribution due to licensing and other restrictions. RPM Fusion is an official extension of the Fedora Project, integrating Livna, Freshrpms, and Dribble.
http://download.fedora.redhat.com	Fedora repository, primary site
http://sources.redhat.com	Open source software hosted by Red Hat
http://torrent.fedoraproject.org	Fedora BitTorrent site for BitTorrent downloads of Fedora distribution ISO images

Table 1-2: Fedora Software Repositories and download sites

fedora

2. Basic System Administration

Superuser Control: the Root User
Fedora Administration Tools
System Directories
Configuration Directories and Files
System Time and Date
Scheduling Tasks: cron
Grand Unified Bootloader (GRUB)
FirstAidKit

Linux is designed to serve many users at the same time, providing an interface between the users and the system with its resources, services, and devices. Users have their own shells through which they interact with the operating system, but you may need to configure the operating system itself in different ways. You may need to add new users, devices like printers and scanners, and even file systems. Such operations come under the heading of system administration. The person who performs such actions is referred to as either a *system administrator* or a *superuser*. In this sense, there are two types of interaction with Linux: regular users' interactions, and those of the superuser, who performs system administration tasks. The chapters in this section cover operations such as changing system runlevels, managing users, installing software, and compiling the kernel. You perform most of these tasks only rarely, such as adding a new printer or mounting a file system. Other tasks, such as adding or removing users, you perform on a regular basis. Basic system administration covers topics such as system access by superusers, selecting the runlevel to start, system configuration files, and system recovery.

With Linux, you have the ability to load different versions of the Linux kernel as well as other operating systems that you have installed on your system. The task of selecting and starting up an operating system or kernel is managed by a boot management utility, the Grand Unified Bootloader (GRUB). This is a versatile tool, letting you load operating systems that share the same disk drive, as well as letting you choose from different Linux kernels that may be installed on the same Linux system.

Terminal Window

The Terminal window allows you to enter Linux commands on a command line. It also provides you with a shell interface for using shell commands instead of your desktop. The command line is editable, allowing you to use the backspace key to erase characters on the line. Pressing any key will insert the key. You can use the left and right arrow keys to move anywhere on the line, and then press keys to insert characters, or use backspace to delete characters (see Figure 2-1).

The terminal window will remember the previous commands you entered. Use the up and down arrow keys to have those commands displayed in turn. Press the ENTER key to re-execute the currently displayed command. You can even edit a previous command before running it, allowing you to execute a modified version of a previous command.

The terminal window will display all your previous interactions and commands for that session. Use the scrollbar to see any previous commands you ran and their displayed results.

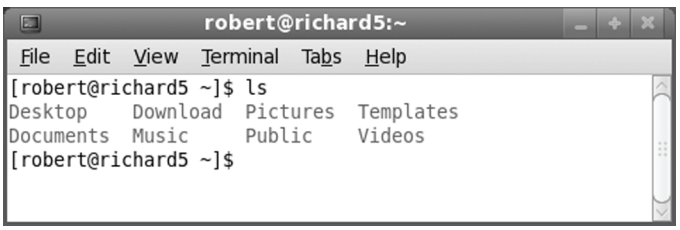


Figure 2-1: Terminal Window

You open as many terminal windows as you want, each working in its own shell. This way you run several command line operations at once, each in their own terminal window using their

own shell. Instead of opening a separate window for each new shell you may want, you can open several shells in the same window, using tabs. Select Open Tab from the File menu to open a new panel, (**Shift-Ctrl-t**). You can use the Tabs menu to move to different tabs, or just click on its tab to select it. Each tab runs a separate shell, letting you enter different commands in each (see Figure 2-2).

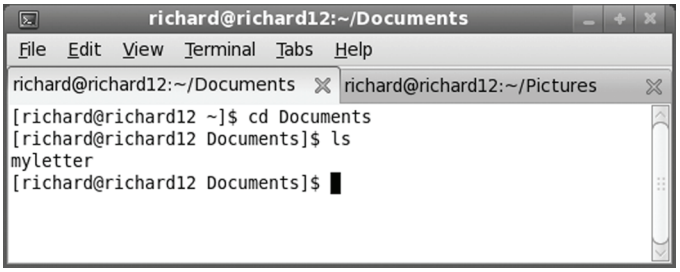


Figure 2-2: Terminal Window with Tabs

The terminal window is also supports GNOME desktop cut/copy and paste operations. You can copy a line from a Web page and then paste it to the terminal window (you may have to use the Copy entry on the Terminal window's Edit menu). The command will appear and then you can press ENTER to execute the command. This is useful for command line operations that may be displayed on an instructional Web page. Instead of typing in a complex command yourself, just copy from the Web page directly, and then paste to the Terminal window. Perform any edits if needed.

Superuser Control: the Root User

To perform system administration operations, you must first have access rights such as the correct password that enables you to log in as the root user, making you the superuser. Because a superuser has the power to change almost anything on the system, such a password is usually a carefully guarded secret, changed very frequently, and given only to those whose job is to manage the system. With the correct password, you can log in to the system as a system administrator and configure the system in different ways. You can start up and shut down the system, as well as change to a different operating mode, such as a single-user mode. You can also add or remove users, add or remove whole file systems, back up and restore files, and even designate the system's name and address.

To become a superuser, you log in to the *root user account.* This is a special account reserved for system management operations with unrestricted access to all components of your Linux operating system. You can log in as the root user from either the GUI (graphical user interface) login screen or the command line login prompt. You then have access to all administrative tools. Using a GUI interface like GNOME, the root user has access to a number of Red Hat and Fedora GUI administrative tool such as PackageKit for installing software or system-config-users for managing users. If you log in from the command line interface, you can run corresponding administrative commands like **`yum`** to install packages or **`useradd`** to add a new user. From your GUI desktop, you can also run command line administrative tools using a terminal window. The command line interface for the root user uses a special prompt, the sharp sign, **#**. In the next example, the user logs in to the system as the root user and receives the **#** prompt.

```
login: root
password:
#
```

Root User Password

As the root user, you can use the **passwd** command to change the password for the root login, as well as for any other user on the system. The **passwd** command will check your password with Pluggable Authentication Modules (PAM), as discussed in Chapter 9, to see if you've selected one that can be easily cracked. To more easily change your root password from a GUI interface, you can use the system-config-rootpassword tool (System | Administration | Root Password).

```
# passwd root
New password:
Re-enter new password:
#
```

You must take precautions to protect your root password. Anyone who gains access as the root user will have complete control over your system. The online manual for the **passwd** command provides detailed recommendations for handling and choosing your password. For example, never store your password in a file on your system, and never choose one based on any accessible information, such as your phone number or date of birth. A basic guideline is to make your password as complex as possible, using a phrase of several words with numbers and upper- and lowercase, yet something you can still remember easily so that you never have to write it down. You can access the **passwd** online manual page with the command

```
# man passwd
```

Root User Access: su

While you are logged in to a regular user account, it may be necessary for you to log in as the root and become a superuser. Ordinarily, you would have to log out of your user account first, and then log in to the root. Instead, you can use the **su** command (switch user) to log in directly to the root while remaining logged in to your user account. If you are using a GUI desktop like GNOME, you can enter the **su** command from a terminal window, or use ALT-CTRL-F6 to switch to a command line interface (ALT-CTRL-F1 returns you to the GUI interface). An **exit** command returns you to your own user login. When you are logged in as the root, you can use **su** to log in as any user, without providing the password. In the next example, the user is logged in already. The **su** command then logs in as the root user, making the user a superuser. Some basic superuser commands are shown in Table 2-1.

```
$ pwd
/home/chris
$ su
 password:
# cd
# pwd
/root
# exit
$
```

When logged in as the **root** user in a terminal window using the **su** command, you will not be able to run any normal GNOME applications as the root user, like the gedit editor. To run normal GNOME applications with root level access, you have to authenticate them individually. For example, the gedit GNOME editor can be run with administrative (root level) access using the **sudo** command, but not as the root user (**su** command). This also means that you cannot login as the root user on the login screen or user switcher (GDM). It is possible to run GNOME administrative tools when logged in as the root user, but it is always preferable to run them as a normal user, where you are then authenticated by being prompted to enter the root user password.

```
sudo gedit
```

Editing is an issue, because you may need to edit system configuration files that are accessible only as the **root** user. You can use the vi editor, if you are familiar with its commands. But many users would prefer a GUI-like editor, such as gedit. You can also use a cursor-based editor like **leafpad** to provide basic GUI-like editing for configuration files when logged in as the root user.

Controlled Administrative Access: sudo

With the sudo tool you can allow ordinary users to have limited root user–level administrative access for certain tasks. This allows other users to perform specific superuser operations without having full root level control. You can find more about sudo at **www.sudo.ws**. To use sudo to run an administrative command, the user precedes the command with the **sudo** command. The user is issued a time-sensitive ticket to allow access.

```
sudo date
```

Access is controlled by the **/etc/sudoers** file. This file lists users and the commands they can run, along with the password for access. If the NOPASSWD option is set, then users will not need a password. ALL, depending on the context, can refer to all hosts on your network, all root-level commands, or all users.

To make changes or add entries, you have to edit the file with the special sudo editing command **visudo**. This invokes the Vi editor to edit the **/etc/sudoers** file. Unlike a standard editor, **visudo** will lock the **/etc/sodoers** file and check the syntax of your entries. You are not allowed to save changes unless the syntax is correct. If you want to use a different editor, you can assign it to the EDITOR shell variable.

```
visudo /etc/sudoers
```

Entries in the sudoers file are technically referred to as specifications, of which there are two kinds: user and alias. The user specification is the primary entry that actually controls user access. The user specification has user, host, runas, and command lists. The user list is the users being controlled in this specification. The hosts are hosts the user can access. The runas list is the list of users the users can access the hosts as (a user with full root user capability should be able to access any user on a host). The command list is the list of commands those users can run. For any of these you can use **ALL** to indicate all users (user list and runas), hosts, or commands. The user specification has the following syntax:

```
users   hosts=(runas) commands
```

The **(***runas***)** segment is optional and is used to allow a user to access a host as another user. The *runas* segment is simply a list of users. Using ALL for the user gives access to all users.

The host is a host on your network. You can specify all hosts with the ALL term. The command can be a list of commands, some or all qualified by options such as whether a password is required. To specify all commands, you can also use the **ALL** term. The following gives the user **george** full root-level access to all commands on all hosts:

```
george  ALL = ALL
```

By default sudo will deny access to all users, including the root. For this reason, the default **/etc/sudoers** file sets full access for the **root** user to all commands. The **ALL=(ALL) ALL** entry allows access by the **root** to all hosts as all users to all commands.

```
root   ALL=(ALL)   ALL
```

To simply give a user full administrative access on all hosts (the same a **root** user access), you can copy the **root** user entry and replace root with the name of the user. The following gives full root user level access to the **richard** user.

```
richard  ALL=(ALL)   ALL
```

When editing the **sudoers** file with the **visudo** editor, to copy the line, move to the **root** line, then press the **yy** keys followed by the **p** key to copy and paste the line. Move to the **root** name in the copy and press the **cw** keys to edit the root name, and then type in the new name. Press the **ESC** key when finished, and then the **ZZ** keys to save and exit

You can also let a user run as another user on a given host. Such alternate users are placed within parentheses before the commands. For example, if you want to give **george** access to the **beach** host as the user **mydns**, you use the following:

```
george beach = (mydns) ALL
```

To let a user run as any user on all hosts, you would use ALL = (ALL), as is done for the root user.

```
george ALL=(ALL) ALL
```

To specify a group name, you prefix the group with a **%** sign, as in **%mygroup**. This way, you can give the same access to a group of users. The **/etc/sudoers** file contains samples for a **%wheel** group.

To give **robert** access on all hosts to the **`date`** command, you would use

```
robert ALL=/usr/bin/system-config-date
```

If a user wants to see what commands he or she can run, that user would use the **`sudo`** command with the **`-l`** option.

```
sudo -l
```

To better control access by different groups and users, sudo also supports alias specifications. See the sudo Man page for more details. You can set up aliases for users, hosts, and commands. Certain aliases are already set up for you. An alias has the following syntax:

```
alias-type   name    list
```

There are three types of aliases, one for each part of a sudo entry: **User_Alias** for users, Runas_Alias for users run as, **Host_Alias** for hosts, and **Cmnd_Alias** for commands. The **Cmnd_Alias** defines groups of commands that can then be easily referenced in a sudo definition.

Several are already defined for you like SERVICES for running the **service** and **chkconfig** commands, and STORAGE for running file system operations like fdisk, parted, and mount.

```
Cmnd_Alias SERVICES = /sbin/service, /sbin/chkconfig
```

Certain **User_Alias** and **Host_Alias** entries are already defined in the **/etc/sudoers** file but commented out, like one for **ADMINS**. These are examples you can use to create an entry of your own.

```
# User_Alias ADMINS = jsmith, mkem
```

The following would allow the user robert to use the service and chkconfig commands on any host.

```
robert ALL=SERVICES
```

If the ADMINS user alias is defined, you could use the following to allow root level access to a group of users designated as administrators.

```
ADMINS  ALL=(ALL)  ALL
```

In addition, sudo also supports tags that provide specialized options. In particular, the **NOPASSWD** tag will allow a user to login without a password. The following would allow a user root level access without having to enter the root password.

```
george ALL = NOPASSWD: ALL
```

Command	Description
su root	Logs a superuser into the root from a user login; the superuser returns to the original login with a CTRL-D.
sudo *command*	Restricted administrative access for specified users.
passwd *login-name*	Sets a new password for the login name.
crontab *options filename*	With *filename* as an argument, installs **crontab** entries in the file to a **crontab** file; these entries are operations executed at specified times (see later section): **-e** Edits the **crontab** file **-l** Lists the contents of the **crontab** file **-r** Deletes the **crontab** file
telinit *runlevel*	Changes the system runlevels.
shutdown *options time*	Shuts down the system.
date	Sets the date and time for the system.
system-config-date	GUI tool to set system time and date (System Settings | Date & Time).
Kcron	KDE GUI interface **cron** management tool (System Tools | Task Scheduler).
system-config-rootpassword	GUI tool to change the root user (administrator) password. (System Settings | Root Password).

Table 2-1: Basic System Administration Tools

There are also options you can set for sudo, including **authenticate** to require a password, and **passwd_timeout** for the password prompt timeout. The **rootpw** option would require that users enter the **root** user password, instead of their own user password.

Fedora Administration Tool	Description
System | Administration	Menu for accessing administrative tools
system-config-users	User and Group configuration tool
system-config-printer	Printer configuration tool
system-config-display	Display configuration tool (video card and monitor)
system-config-services	Service management
system-config-rootpassword	Changes the root user password
system-config-keyboard	Changes the keyboard configuration
system-config-date	Changes system time and date
system-config-time	Changes system time and date
system-config-language	Selects a language to use
system-config-vsftpd	Manage the vsftpd FTP server
system-config-selinux	Manage and configure SELinux
PackageKit	Software management (PolicyKit).
GNOME International clock applet	Change time and date, display weather (PolicyKit)
system-config-network	Fedora network configuration tool for all types of connections.
Network Manager	Automates wireless and standard network connection selection and notification.
KNetworkManager	KDE tool to automate wireless and standard network connection selection and notification.
system-control-network	Fedora network device control tool for turning network connections on an off.
system-config-services	Starts and stops servers, including network servers (smb for Samba, httpd for Web, bind for DNS, and nfs for NFS).
system-config-firewall	Sets up a network firewall.
system-config-bind	Configures a domain name server.
system-config-samba	Configures Samba shares.
system-config-nfs	Configures NFS shares.
system-config-httpd	Configures an Apache Web server.
system-config-netboot	Configures diskless workstations and network installation.

Table 2-2: Fedora Configuration Tools

Fedora Administration Tools

On Fedora, most administration tasks can be handled by a set of separate specialized administrative tools developed and supported by Red Hat and Fedora, such as those for user management and printing configuration. Many of these are GUI-based and will work on any X Window System environment, such as GNOME or KDE. You can access the GUI-based Fedora tools as any user from the GNOME System | Administration menu. Here you will find tools to set the time and date, manage users, configure printers, and update software. Users and Groups lets you create and edit users. Printing lets you install and reconfigure printers. All tools provide very intuitive GUI interfaces that are easy to use. Tools are identified by simple descriptive terms, whereas their actual name normally begins with the term **system-config**. For example, the printer configuration tool is listed as Printing, but its actual name is system-config-printer. You can separately invoke any tool by entering its name in a terminal window.

For most tools requiring root level access, the program name is actually a link to the **consolehelper** tool which will first invoke PAM authentication rules to prompt for administrative access. Each system-config administrative tool has its own PAM configuration file in **/etc/pam.d**. These will include the **config-util** file which will specify the PAM authentication modules to use. Fedora does not use PolicyKit to manage system-config tools. There is one major exception. PackageKit, the software management tool, is controlled by PolicyKit, not PAM. Also, the GNOME International Clock applet is also controlled by PolicyKit

Table 2-2 provides a listing of Red Hat and Fedora administration tools.

System Directories

Your Linux file system is organized into directories whose files are used for different system functions (see Table 2-3). For basic system administration, you should be familiar with the system program directories where applications are kept, the system configuration directory (**/etc**) where most configuration files are placed, and the system log directory (**/var/log**) that holds the system logs, recording activity on your system. Other system directories are covered in their respective chapters.

Program Directories

Directories with "bin" in the name are used to hold programs. The **/bin** directory holds basic user programs, such as login shells (BASH, TCSH, and ZSH) and file commands (**cp**, **mv**, **rm**, **ln**, and so on). The **/sbin** directory holds specialized system programs for such tasks as file system management (**fsck**, **fdisk**, **mkfs**) and system operations like shutdown and startup (**init**). The **/usr/bin** directory holds program files designed for user tasks. The **/usr/sbin** directory holds user-related system operations, such as **useradd** to add new users. The **/lib** directory holds all the libraries your system makes use of, including the main Linux library, **libc**, and subdirectories such as **modules**, which holds all the current kernel modules.

Configuration Directories and Files

When you configure different elements of your system, like users, applications, servers, or network connections, you make use of configuration files kept in certain system directories. Configuration files are placed in the **/etc** directory, with more specific device and service configuration located in the **/etc/sysconfig** directory.

Directories	Description
/bin	System-related programs
/sbin	System programs
/usr/sbin	System programs for user related tasks
/lib	System and application libraries
/etc	Configuration files for system and network services and applications
/home	The location of user home directories and server data directories, such as Web and FTP site files
/mnt	The location where CD-ROM and floppy disk file systems are mounted)
/var	The location of system directories whose files continually change, such as logs, printer spool files, and lock files
/usr	User-related programs and files. Includes several key subdirectories, such as **/usr/bin**, **/usr/X11**, and **/usr/doc**
/usr/bin	Programs for users
/dev	Dynamically generated directory for device files
/etc/X11	X Window System configuration files
/usr/share	Shared files
/usr/share/doc	Documentation for applications
/tmp	Directory for system temporary files
/var/log	Logging directory
/var/log/	System logs generated by **rsyslogd**
/var/log/audit	Audit logs generated by **auditd**

Table 2-3: System Directories

Note: If SELinux is enabled super user access will be controlled by SELinux rules. See Chapter 12.

Configuration Files: /etc

The **/etc** directory holds your system, network, server, and application configuration files. Here you can find the **fstab** file listing your file systems, the **hosts** file with IP addresses for hosts on your system, and **grub.conf** (a link to **/boot/grub/grub.conf**) for the boot systems supported by the GRUB boot loader. This directory includes various subdirectories, such as **/etc/apache** for the Apache Web server configuration files, **/etc/X11** for the X Window System and window manager configuration files, and **/etc/udev** for rules to generate device files in **/dev**. You can configure many applications and services by directly editing their configuration files, though it is best to use a corresponding administration tool. Table 2-4 lists several commonly used configuration files found in the **/etc** directory.

File	Description
/etc/bashrc	Default shell configuration file Bash shell
/etc/group	Contains a list of groups with configurations for each
/etc/fstab	Automatically mounts file systems when you start your system
/boot/grub/grub.conf	The GRUB configuration files for the GRUB boot loader, linked to by **/etc/grub.conf**.
/etc/inittab	Sets the default state, as well as terminal connections
/etc/profile	Default shell configuration file for users
/etc/motd	System administrator's message of the day
/etc/mtab	Currently mounted file systems
/etc/passwd	Contains user password and login configurations
/etc/services	Services run on the system and the ports they use
/etc/shadow	Contains user-encrypted passwords
/etc/shells	Shells installed on the system that users can use
/etc/sudoers	Sudo configuration to control administrative access
/etc/xinetd.conf	Xinetd server configuration
Directories	
/etc/cron	Cron scripts
/etc/cups	CUPS printer configuration files
/etc/init.d	Service scripts
/etc/modprobe.d	Directory with kernel module configuration files specifying modules on your system to be automatically loaded
/etc/openldap	Configuration for Open LDAP server
/etc/rc.d	Startup scripts for different runlevels
/etc/skel	Directory that holds the versions of initialization files, such as **.bash_profile**, which are copied to new users' home directories
/etc/sysconfig	Red Hat and Fedora device and service configuration environments
/etc/terminfo	Contains terminal type specifications for terminals that could be connected to the system
/etc/X11	X Window System configuration files
/etc/xinetd.d	Configuration scripts for services managed by Xinetd server
/etc/udev	Rules for generating devices
/etc/hal	Rules for generating removable devices

Table 2-4: Configuration Files and Directories

/etc/sysconfig

On Red Hat and Fedora systems, configuration and startup information is also kept in the **/etc/sysconfig** directory. Here you will find files containing definitions of system variables used to configure devices such as your keyboard and mouse, along with settings for network connections, as well as options for service scripts, covering services such as the Web server or the IPtables firewall. These entries were defined for you when you configured your devices during installation or installed your service software.

A sample of the keyboard file, **/etc/sysconfig/keyboard**, is shown here.

```
KEYBOARDTYPE="pc"
KEYTABLE="us"
```

Several of these files are generated by Fedora administration tools such as system-config-firewall or system-config-network**.** Table 2-5 lists several commonly used tools and the sysconfig files they control.

Some files provide global or system configuration support for service scripts, like **iptables**, **samba**, **httpd** (Apache), or **SpamAssasin**. Other files provide configuration settings for corresponding tools like system-config-users. Several directories are included, such as **network-scripts**, which list several startup scripts for network connections—an example is **ifup-ppp**, which starts up PPP connections.

Tools	Configuration Files	Description
system-config-authentication	**/etc/sysconfig/authconfig /etc/sysconfig/network**	Authentication options, such as enabling NIS, shadow passwords, Kerberos, and LDAP.
system-config-keyboard	**/etc/sysconfig/keyboard**	Selects the keyboard type.
system-config-network	**/etc/sysconfig/network /etc/sysconfig/networking**	Sets your network settings.
system-config-date	**/etc/sysconfig/clock**	Sets the time and date.
system-config-users	**/etc/sysconfig/system-config-users**	Settings for system-config-users.
system-config-samba	**/etc/sysconfig/samba**	Settings for Samba service.
system-config-httpd	**/etc/sysconfig/httpd**	Settings for Apache Web server.
system-config-firewall	**/etc/sysconfig/system-config-firewall**	Settings for system-config-firewall.

Table 2-5: Sysconfig Files with Corresponding Fedora System Administration Tools

Some administration tools use more than one **sysconfig** file; for example, system-config-network places its network configuration information such as the hostname and gateway in the **/etc/sysconfig/network** file. Specific Ethernet device configurations, which would include your IP address and netmask, are placed in the appropriate Ethernet device configuration file in the **/etc/sysconfig/network-scripts** directory. For example, the IP address and netmask used for the **eth0** Ethernet device can be found in **/etc/sysconfig/network-scripts/ifcfg-eth0**. Local host settings are in **/etc/sysconfig/network-scripts/ifcfg-lo**.

Network configuration set up by **system-config-network** can be found in the **/etc/sysconfig/networking** directory. The devices subdirectory holds the ifcfg file for you **device**, whereas the **profiles** directory holds the profiles you may have set up with **system-config-network**. Here you will find the host and nameserver information for your network connection profiles, including the default profile.

Tip: Some administration tools, like system-config-authentication, will further configure configuration files for the services selected. The system-config-authentication tool configures **/etc/sysconfig/authconfig**, as well as **/etc/krb5.conf** for Kerberos authentication, **/etc/yp.conf** for NIS support, and **/etc/openldap/ldap.conf** for LDAP authentication.

System Time and Date

You can set the system time and date using the shell **`date`** command, the GNOME clock applet, or the Fedora GUI tool system-config-date. You probably set the time and date when you first installed your system. You should not need to do so again. If you entered the time incorrectly or moved to a different time zone, though, you could use this utility to change your time.

GNOME International Clock: Time, Date, and Weather

The GNOME international clock applet is located on the top panel to the right. It displays the current time and date for your region, but can be modified to display the weather, as well as the time, date, and weather of any location in the world.

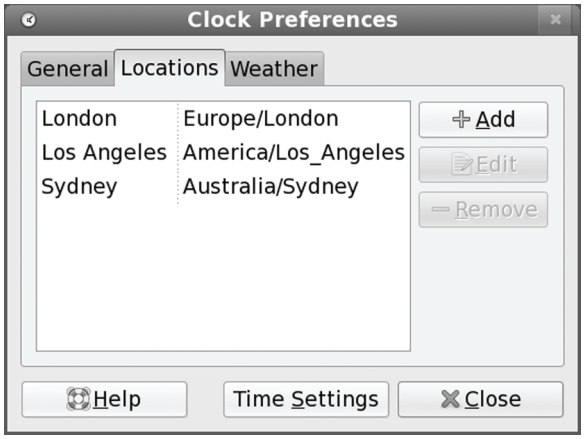


Figure 2-3: Selecting a location on the international clock

To add a location, right-click on the time and select Preferences from the pop-up menu. The Clock Preferences window will display three panels: General, Location, and Weather. To add a new location, click on the Add button on the Location panel (see Figure 2-3). This opens a box where you can enter the name, timezone, and coordinates of the location. It is easier to just click the Find button and open an expandable tree of locations, starting with continent, then to region, country, and city.

On the Weather panel, you can specify the temperature and wind measures to use. The General panel you can set the clocks display options for the locations, whether to show weather, temperature, date and seconds.

Once you set the location for your own location, you will see a weather icon appear next to the time on the panel, showing you your current weather (see Figure 2-4). The icon will change depending on the type of weather (cloudy, raining, overcast, clear, etc).

Figure 2-4: International clock with weather icon

To see the locations you have selected, click on the time displayed on the top panel (see Figure 2-5). This opens a calendar, with Location label with an expandable arrow at the bottom.

Click this arrow to display all your locations, their time and weather. You home location will have a house icon next to it. A world map will show all your locations as red dots, with a blue house icon for your current home location. When you click on a location entry, its corresponding dot will blink for a few seconds. Each location will have a small globe weather icon, indicating the general weather, like sun or clouds. To see weather details, move your mouse over the weather icon. A pop-up dialog will display the current weather, temperature, wind speed, and time for sunrise and sunset. The clock icons for each location will be dark, grey, or bright depending on the time of day at that location.

Figure 2-5: International clock, full display

You can easily change your home location, by click the Set button to the right the location you want made your home. You will see the home icon shift to the new location. This is helpful when traveling. Each location has a set button which will be hidden until you move your mouse over it, on the right side of the clock display

To make any changes, you can click the Edit button next to the Locations label. This opens the Clock Preferences window where you can configure the display or add and remove locations.

The calendar will show the current date, but you can move to different months and years using the month and year scroll arrows at the top of the calendar.

To set the time manually, right-click on the time and select Adjust Date & Time (also from the General panel of the Clock Preferences window, click the Time Settings button). This opens a Time Settings window where you can enter the time and set the date (see Figure 2-6). Use the month and year arrows on the calendar to change the month and year.

To set the time for the entire system, click the Set System Time button. You will be prompted to authenticate by entering your password.

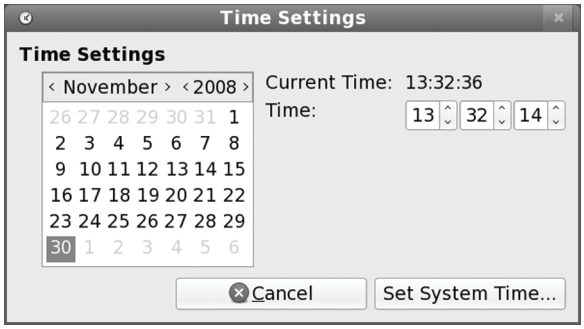


Figure 2-6: Manually setting the clock

Using the system-config-date Utility

The preferred way to set the system time and date is to use the Fedora Date and Time Properties utility (system-config-date or system-config-time). Choose the System | Administration | Date & Time entry on the GNOME System menu. There are three panels, one for the date and time, one for the Network Time Protocol, and one for the time zone (see Figure 2-7). Use the calendar to select the year, month, and date. Then use the Time box to set the hour, minute, and second. The Time Zone panel shows a map with locations. Select the one nearest you to set your time zone.

The Network Time Protocol (NTP) allows a remote server to set the date and time, instead of using local settings. NTP allows for the most accurate synchronization of your system's clock. It is often used to manage the time and date for networked systems, freeing the administrator from having to synchronize clocks manually. You can download current documentation and NTP software from the **www.ntp.org** site.

On the Network Time Protocol panel you can choose to enable NTP and select the server to use. NTP servers operate through pools which will randomly select an available server to increase efficiency. A set of pools designated for use by Fedora are already installed up for you, beginning with **0.fedora.pool.ntp.org**. If access with one pool is slow, you change to another. The pool servers support worldwide access. Pools for specific geographical locations can be found at the NTP Public Services Project site (Time Servers link), **http://ntp.isc.org**. A closer server could be faster.

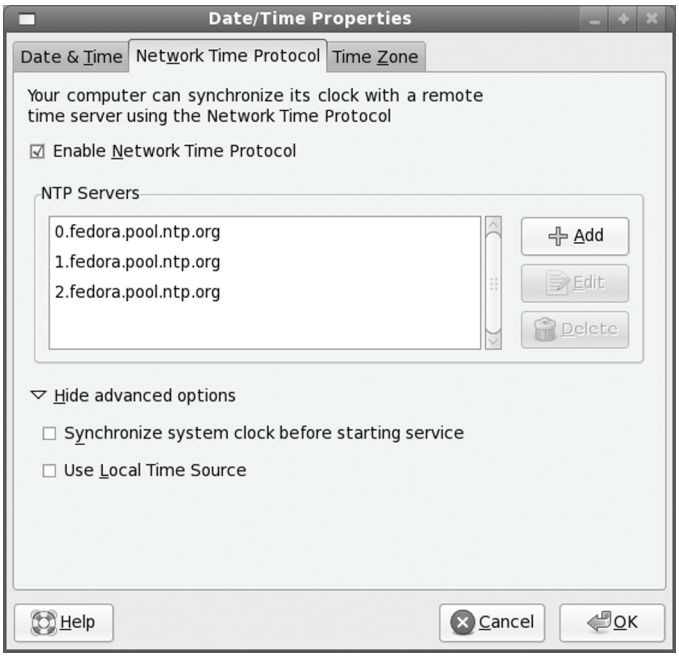


Figure 2-7: system-config-date

Using the date Command

You can use the **date** command on your root user command line to set the date and time for the system. As an argument to **date**, you list (with no delimiters) the month, day, time, and year. In the next example, the date is set to 2:59 P.M., April 6, 2008 (04 for April, 06 for the day, 1459 for the time (2:59 PM), and 08 for the year 2008):

```
date 0406145908
Sun Apr 6 02:59:22 PST 2008
```

Scheduling Tasks: cron

Scheduling regular maintenance tasks, such as backups, is managed by the **cron** service on Linux, and implemented by a **cron** daemon. A daemon is a continually running server that constantly checks for certain actions to take. These tasks are listed in the **crontab** file. The **cron** daemon constantly checks the user's **crontab** file to see if it is time to take these actions. Any user

can set up a **crontab** file of his or her own. The root user can set up a **crontab** file to take system administrative actions, such as backing up files at a certain time each week or month.

The easiest way to schedule tasks is to use the desktop **cron** tool. For KDE you can use the KCRON tool and for GNOME you can use GNOME Schedule. Both provide panels for choosing the month, date, and time for a process, though you will have to manually enter the command you want run, as if on a command line. A listing of **cron** entries lets you modify or delete tasks. If you have an open ended operation, be sure to also schedule a command to shut it down.

GNOME Schedule

GNOME Schedule is a more recent tool that also creates and easy to use interface for managing scheduled tasks (See Figure 2-8). Once installed you can access it from the Applications | System Tools menu as Scheduled Tasks. The GNOME Schedule command features templates where you can repeat a task, setting a different time, without have to enter the entire task again. Very useful for complex commands.

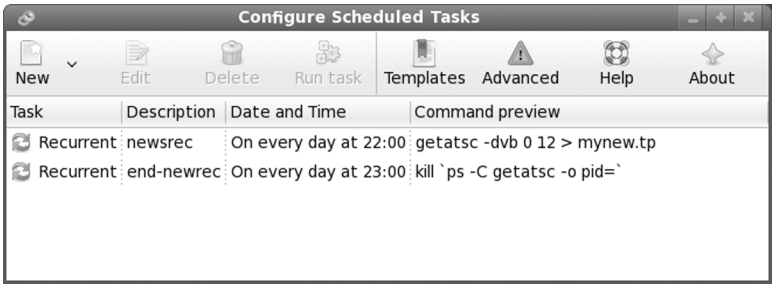


Figure 2-8: GNOME Schedule

Use the New button to schedule a task. You are first asked if you want to create the task as a recurrent item, one time task, or from a template. The Create a New Scheduled Task window then open where you can specify the time and date, and whether to repeat weekly or monthly (see Figure 12-8). You can use the Basic button to set defaults for Hourly, Daily, Weekly, or Monthly entries. Then click Advanced to specify a time.

The template feature lets you set up a new schedule with information from a previous one, using the same or similar commands but different time. Click the Templates button to add a new template. This opens a window similar to the Create task window in Figure 2-9. You can also create a template from a current task. Select the task and click the Edit button. On the Edit window of that task, click the "Add as template" button.

Once you have created the template you can use it to create scheduled tasks. When creating a task, from the initial menu, choose "A task from a predefined template". This opens the Choose template window (see Figure 2-10). Select the template to use. Clicking the Use template button opens the Create task window where you can modify your task, changing its name or time as needed, and then clicking the Apply button to create the task.

To delete a task, just select the entry in the Scheduled Tasks window and click the Delete button. To run a task immediately, select the task and click the Run task button.

On the Scheduled Tasks window you can click the Advanced button to see the actual cron entries created by GNOME Schedule.

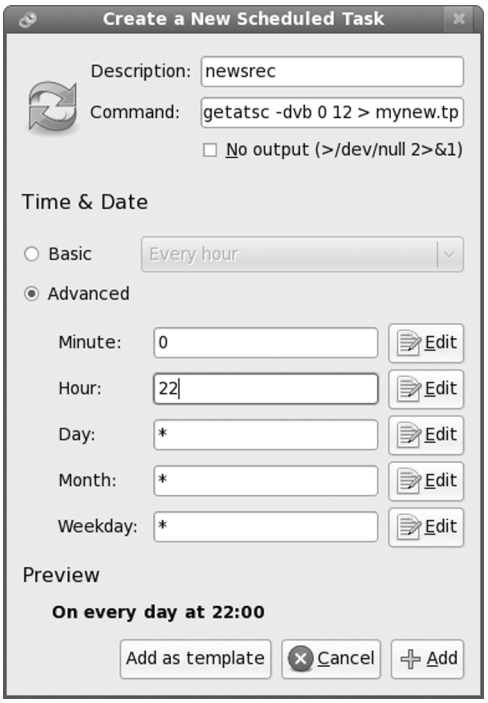


Figure 2-9: GNOME Schedule new task

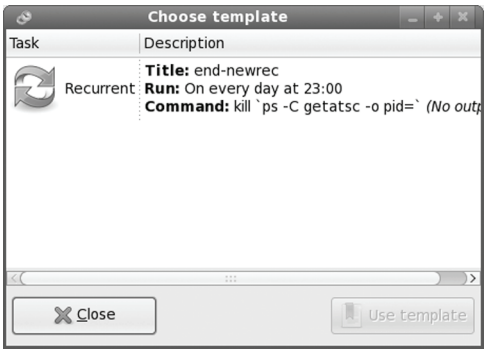


Figure 2-10: GNOME Schedule templates

KCron (KDE4)

KCron which creates and easy to use interface for creating scheduled commands. KCron is a KDE desktop tool and will require installation of supporting KDE libraries (selected automatically for you when you install KCron). To run KCron, login to the KDE desktop. Use the

New entry in the Edit menu to schedule a command, specifying the time and date, and whether to repeat weekly or monthly.

The crond Service

The name of the **cron** daemon is crond. Normally it is started automatically when your system starts up. You can set this feature using system-config-services or **chkconfig**, as described in Chapter 3. The following example starts the **crond** service automatically whenever you boot the system.

```
chkconfig crond on
```

You can also start and stop the **crond** service manually, which you may want to do for emergency maintenance or during upgrades. Use the **service** command and the **stop** option to shut down the service, and the **start** option to run it again, and **restart** to restart it:

```
service crond stop
```

crontab Entries

Cron task schedules are kept in **crontab** files, like **/etc/crontab**. An entry has six fields: the first five are used to specify the time for an action, while the last field is the action itself. The first field specifies minutes (0–59), the second field specifies the hour (0–23), the third field specifies the day of the month (1–31), the fourth field specifies the month of the year (1–12, or month prefixes like *Jan* and *Sep*), and the fifth field specifies the day of the week (0–6, or day prefixes like *Wed* and *Fri*), starting with 0 as Sunday. In each of the time fields, you can specify a range, a set of values, or use the asterisk to indicate all values. For example, **1-5** for the day-of-week field specifies Monday through Friday. In the hour field, **8, 12, 17** would specify 8 A.M., 12 noon, and 5 P.M. An ***** in the month-of-year field indicates every month. The format of a cron field follows:

```
minute  hour  day-month  month  day(s)-week  task
```

The following example backs up the **projects** directory at 2:00 A.M. every weekday:

```
0 2 * * 1-5   tar cf /home/backp /home/projects
```

The same entry is listed here again using prefixes for the month and weekday:

```
0 2 * * Mon-Fri tar cf /home/backp /home/projects
```

To specify particular months, days, weeks, or hours, you can list them individually, separated by commas. For example, to perform the previous task on Sunday, Wednesday, and Friday, you could use **0,3,5** in the day-of-week field, or their prefix equivalents, **Sun,Wed,Fri**.

```
0 2 * * 0,3,5   tar cf /home/backp /home/projects
```

Cron also supports comments. A comment is any line beginning with a **#** sign.

```
# Weekly backup for Chris's projects
0 2 * * Mon-Fri  tar cf /home/backp /home/projects
```

Environment Variables for cron

The **cron** service also lets you define environment variables for use with tasks performed. Fedora defines variables for **SHELL**, **PATH**, **HOME**, and **MAILTO**. **SHELL** designates the shell to use tasks, in this case the bash shell. **PATH** lists the directories where programs and scripts can be found. This example lists the standard directories, **/usr/bin** and **/bin**, as well as the system directories reserved for system applications, **/usr/sbin** and **/sbin**. **MAILTO** designates to who the results of a task are to be mailed. By default, these are mailed to the user who schedules it, but you can have the results sent to a specific user, such as the administrator's e-mail address, or an account on another system in a network. **HOME** is the home directory for a task, in this case the top directory.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
```

The cron.d Directory

On a heavily used system, the **/etc/crontab** file can become crowded easily. There may also be instances where certain entries require different variables. For example, you may need to run some task under a different shell. To help better organize your cron tasks, you can place entries in files within the **cron.d** directory. The files in the **cron.d** directory all contain entries of the same format as those used in **/etc/crontab**. They may be given any name. They are treated as added **crontab** files, with **cron** checking them for tasks to run. For example, Fedora installs a **smolt** file in the **cron.d** that contains entries to run tools to gather hardware profiles.

The crontab Command

You use the **crontab** command to install your entries into a **crontab** file. To do this, first create a text file and type your **crontab** entries. Save this file with any name you want, such as **mycronfile**. Then, to install these entries, enter **crontab** and the name of the text file. The **crontab** command takes the contents of the text file and creates a **crontab** file in the **/var/spool/cron** directory, adding the name of the user who issued the command. In the following example, the root user installs the contents of **mycronfile** as the root's **crontab** file. This creates a file called **/var/spool/cron/root**. If a user named justin installed a **crontab** file, it would create a file called **/var/spool/cron/justin**. You can control use of the **crontab** command by regular users with the **/etc/cron.allow** file. Only users with their names in this file can create **crontab** files of their own. Conversely, the **/etc/cron.deny** file lists those users denied use of the **cron** tool, preventing them for scheduling tasks. If neither file exists, access is denied to all users. If a user is not in a **/etc/cron.allow** file, access is denied. However, if the **/etc/cron.allow** file does not exist, and the **/etc/cron.deny** file does, then all users not listed in **/etc/cron.deny** are automatically allowed access.

```
# crontab mycronfile
```

Editing in cron

Never try to edit your **crontab** file directly. Instead, use the **crontab** command with the **-e** option. This opens your **crontab** file in the **/var/spool/cron** directory with the standard text editor, such as **vi**. **crontab** uses the default editor as specified by the **EDITOR** shell environment

variable. To use a different editor for **crontab**, change the default editor by assigning the editor's program name to the **EDITOR** variable and exporting that variable. Normally, the editor variable is set in the **/etc/profile** script.

Running **crontab** with the **-l** option displays the contents of your **crontab** file, and the **-r** option deletes the entire file. Invoking **crontab** with another text file of **crontab** entries overwrites your current **crontab** file, replacing it with the contents of the text file.

Organizing Scheduled Tasks

You can organize administrative **cron** tasks into two general groups: common administrative tasks that can be run at regular intervals, or specialized tasks that need to be run at a unique time. Unique tasks can be run as entries in the **/etc/crontab** file, as described in the next section. Common administrative tasks, though they can be run from the **/etc/crontab** file, are better organized into specialized **cron** directories. Within such directories, each task is placed in its own shell script that will invoke the task when run. For example, there may be several administrative tasks that all need to be run each week on the same day, say if maintenance for a system is scheduled on a Sunday morning. For these kinds of task, **cron** provides several specialized directories for automatic daily, weekly, monthly, and yearly tasks. Each contains a **cron** prefix and a suffix for the time interval. The **/etc/cron.daily** directory is used for tasks that need to be performed every day, whereas weekly tasks can be placed in the **/etc/cron.weekly** directory. The **cron** directories are listed in Table 2-6.

cron Files and Directories	Description
/etc/crontab	System **crontab** file, accessible only by the root user
/etc/cron.d	Directory containing multiple **crontab** files, accessible only by the root user
/etc/cron.hourly	Directory for tasks performed hourly
/etc/cron.daily	Directory for tasks performed daily
/etc/cron.weekly	Directory for tasks performed weekly
/etc/cron.monthly	Directory for tasks performed monthly
/etc/cron.yearly	Directory for tasks performed yearly
/etc/cron.hourly	Directory for tasks performed hourly
/etc/cron.allow	Users allowed to submit **cron** tasks
/etc/cron.deny	Users denied access to **cron**

Table 2-6: Cron Files and Directories

Running cron Directory Scripts

Each directory contains scripts that are all run at the same time. The scheduling for each group is determined by an entry in the **/etc/crontab** file. The actual execution of the scripts is performed by the **/usr/bin/run-parts** script, which runs all the scripts and programs in a given

directory. Scheduling for all the tasks in a given directory is handled by an entry in the **/etc/crontab** file. Fedora provides entries with designated times, which you may change for your own needs. The default Fedora **crontab** file is shown here, with times for running scripts in the different **cron** directories. Here you can see that most scripts are run at about 4 A.M. either daily (4:02), Sunday (4:22), or first day of each month (4:42). Hourly ones are run one minute after the hour.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

Tip: Scripts within a **cron** directory are run alphabetically. If you need a certain script to run before any others, you may have to alter its name. One method is to prefix the name with a numeral. For example, in the **/cron.weekly** directory, the **anacron** script is named **0anacron** so that it will run before any others.

Keep in mind though that these are simply directories that contain executable files. The actual scheduling is performed by the entries in the **/etc/crontab** file. For example, if the weekly field in the **`cron.weekly` `crontab`** entry is entry is changed to * instead of **`0`**, and the monthly field to **`1`** (**`22 4 1 * *`** instead of **`22 4 * * 0`**), tasks in the **cron.weekly** file would end up running monthly instead of weekly.

Cron Directory Names

The names used for these directories are merely conventions. They have no special meaning to the **cron** daemon. You could, in fact, create your own directory, place scripts within it, and schedule **run-parts** to run those scripts at a given time. In the next example, scripts placed in the **/etc/cron.mydocs** directory will run at 12 noon every Wednesday.

```
* 12 * * 3 root run-parts /etc/cron.mydocs
```

Anacron

For a system that may normally be shut down during times that **cron** is likely to run, you may want to supplement **cron** with **anacron**. **anacron** activates only when scheduled tasks need to be executed. For example, if a system is shut down on a weekend when **cron** jobs are scheduled, then the jobs will not be performed; **anacron**, however, checks to see what jobs need to be performed when the system is turned on again, and then runs them. It is designed only for jobs that run daily or weekly.

For **anacron** jobs, you place **`crontab`** entries in the **/etc/anacrontab** file. For each scheduled task, you specify the number of intervening days when it is executed (7 is weekly, 30 is monthly), the time of day it is run (numbered in minutes), a description of the task, and the command to be executed. For backups, the command used would be **tar** operation. You can use system-config-services to turn on the **anacron** service or have it start up automatically at boot time.

Grand Unified Bootloader (GRUB)

The Grand Unified Bootloader (GRUB) is a multiboot boot loader used for Fedora and Red Hat. With Grub, users can select operating systems to run from a menu interface displayed when a system boots up. Use arrow keys to move to an entry and press ENTER. Linux and Unix operating systems are known as multiboot operating systems and take arguments passed to them at boot time. Type **e** to edit a command, letting you change kernel arguments or specify a different kernel. The **c** command places you in a command line interface. Provided your system BIOS supports very large drives, GRUB can boot from anywhere on them. Check the grub Man page for Grub options. You can use the Boot Configuration tool to select your default system or kernel as well as set the timeout limit (accessible from System | Administration menu as Bootloader).

GRUB configuration is held in the **/boot/grub/grub.conf** file (You can also access this file as **/etc/grub.conf** which is link to **/boot/grub/grub.conf**). You only need to make your entries and GRUB will automatically read them when you reboot. There are several options you can set, such as the timeout period and the background image to use. Check the Grub info documentation for a detailed description, **info grub**.

You can boot to different kernels as well as to different operating systems. Kernel entries are place in the **grub.conf** file automatically when you install or update your system update adds a new kernel. The grub.conf file is also automatically updated should you add a kernel manually through YUM.

Should you want to manually create a new entry for a specific operating system, you would edit the **/boot/grub/grub.conf** file to add in the entry. You first specify a system to boot by creating a title entry for it, beginning with the term **title**. You then have to specify where the operating system kernel or program is located, which hard drive and what partition on that hard drive. This information is listed in parentheses following the **root** option. Numbering starts from 0, not 1, and hard drives are indicated with an **hd** prefix, whether they are IDE, SATA, or SCSI hard drives. So **root(hd0,2)** references the first hard drive (**sda**) and the third partition on that hard drive (**sda3**). For Linux systems, you will also have to use the **kernel** option to indicate the kernel program to run, using the full pathname and any options the kernel may need. The RAM disk is indicated by the **initrd** option.

```
title Fedora Linux (2.6.27.5-117.fc10.x86_64)
      root (hd0,2)
      kernel /boot/vmlinuz-2.6.27.5-117.fc10.x86_64 ro root=/dev/sda3 rhgb quiet
      initrd /boot/initrd-2.6.27.5-117.fc10.x86_64.img
```

Though you still reference a partition with the by hard drive device name and partition numbers, the kernel entry will invoke the kernel using the hard drive partition's LVM volume name (if on an LVM volume) or the hard drive's UUID name. Examples of both are shown here (myfedora10 is an LVM group):

```
kernel /boot/vmlinuz-2.6.27.5-117.fc10.x86_64 ro root=UUID=33f43598-65c0-44c8-
b1e3-a3dedb61151cf rhgb quiet

kernel /boot/vmlinuz-2.6.27.5-117.fc10.x86_64 ro root=/dev/myfedora10/myfedora1
rhgb quiet
```

The kernel option specifies the kernel to run. The kernel is located in the /boot directory and has the name **vmlinuz** with the kernel version number. You can have several kernels in the

/boot directory and use grub to choose the one to use. After the kernel program you specify any options you want for the kernel. This includes a **ro** option which initially starts the kernel as read only. The root option is used to specify the device on which your system was installed, your root directory. In the previous example the system was installed on device **/dev/sda3**. If the root directory is on a logical volume, where it is normally installed, the device name would be something like **/dev/VolGroup00/LogVol01**. The **rhgb quiet** options uses the Red Hat Graphical Boot interface to start your system, instead of listing all the start up messages on a text display.

Note: If you are having start up problems with your graphic card driver, you may need to remove **rhgb quiet** from your kernel option in grub.conf. This starts up Linux using the command line interface.

If you installed the standard Workstation configuration, your root directory will be installed on a logical volume. Your root option would reference the logical volume, specifying the volume group and the logical volume, as shown here.

```
kernel /vmlinuz-2.6.27.5-117.fc10.x86_64 ro root=/dev/VolGroup00/LogVol00 rhgb
quiet
```

For another operating system such as Windows, you would use the **rootnoverify** option to specify where Windows is installed. This option instructs GRUB not to try to mount the partition. Use the **chainloader+1** option to allow GRUB to access it. The **chainloader** option tells GRUB to use another boot program for that operating system. The number indicates the sector on the partition where the boot program is located—for example, **+1** indicates the first sector.

```
title Windows XP
      rootnoverify (hd0,0)
      chainloader +1
```

Windows systems will all want to boot from the first partition on the first disk. This becomes a problem if you want to install several versions of Windows on different partitions or install Windows on a partition other than the first one. GRUB lets you work around this by letting you hide other partitions in line, and then unhiding the one you want, making it appear to be the first partition. In this example, the first partition is hidden, and the second is unhidden. This assumes there is a Windows system on the second partition on the first hard drive (**hd0,1**). Now that the first partition is hidden, the second one appears as the first partition:

```
hide (hd0,0)
unhide (hd0,1)
rootnoverify (hd0,1)
```

To boot a Windows system from a hard drive that is not the first hard drive, you can use the map operation to make the other hard drive the first hard drive. In the following example, the first and second hard drives switch position. The Window operating system is located on the second hard drive, first partition **(hd1,0)**. Even when switched, you still reference its original position with the rootnoverify command, **(hd1,0) +1**.

```
map (hd0)(hd1)
map (hd1)(hd0)
rootnoverify (hd1,0) +1
```

A sample **grub.conf** file follows with entries for both Linux and Windows. Notice that kernel parameters are listed in the **`kernel`** option as arguments to the kernel. The root directory is installed on a logical volume, **/dev/VolGroup00/LogVol01**.

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition. This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,1)
#          kernel /vmlinuz-version ro root=/dev/VolGroup00/LogVol01
#          initrd /initrd-version.img
#boot=/dev/sda
default=1
timeout=5
splashimage=(hd0,1)/grub/splash.xpm.gz
hiddenmenu


title Fedora (2.6.27.5-117.fc10.x86_64)
  root (hd0,1)
  kernel /vmlinuz-2.6.27.5-117.fc10.x86_64 ro root=/dev/VolGroup00/LogVol01 quiet
  initrd /initrd-2.6.27.5-117.fc10.x86_64.img


title Windows XP
  rootnoverify (hd0,0)
  chainloader +1
```

Tip: If you have problems booting to Fedora, and you can fix the issue by editing the **grub.conf** file (like changing hard drive numbers), you can boot up with your CD/DVD Fedora install disk and type **linux rescue** at the boot prompt. Follow the prompts to boot up your system with the command line interface. Issue the **chroot /mnt/sysimage** command. You can then change to the **/boot/grub** directory and edit your **grub.conf** file with an editor like **vi**.

Rescue

If for some reason you are not able to boot or access your system, it may be due to conflicting configurations, libraries, or applications. In this case, you can boot your Linux system in a rescue mode and then edit configuration files with a text editor such as Vi, remove the suspect libraries, or reinstall damaged software with RPM. To enter the rescue mode, CD-ROM or the DVD-ROM boot disk, select **Rescue Installed System** on the initial menu.

You will boot into the command line mode with your system's files mounted at **/mnt/sysimage**. You will be notified that you can use the **`chroot`** command to set your system to the / directory as the root directory. Issue the following command at the command line prompt:

```
chroot /mnt/sysimage
```

Use the **cd** command to move between directories. Check **/etc** and **/etc/sysconfig** for your configuration files. You can use Vi to edit your files and the **`less`** command to view them. To reinstall files, use the **`rpm`** command. When you are finished, enter the **`exit`** command.

If you have a command line system, at the boot prompt, enter

```
linux rescue
```

You can also download and create the Linux rescue CD. Just download the Linux rescue CD iso from the Fedora repository and boot with this disk.

Re-installing the boot loader

If you have a dual-boot system, where you are running both Windows and Linux on the same machine, you may run into a situation where you have to re-install your GRUB boot loader. This problem occurs if your Windows system completely crashes beyond repair and you have to install a new version of Windows, or you are adding Windows to your machine after having installed Linux. Windows will automatically overwrite your boot loader (alternatively, you could install your boot loader on your Linux partition instead of the MBR). You will no longer be able to access your Linux system.

All you need to do is to reinstall your boot loader. First boot from your Linux DVD/CD-ROM installation disk, and at the menu select **Rescue Installed System**.

As noted in the preceding section, this boots your system in rescue mode. Then use **`grub-install`** and the device name of your first partition to install the boot loader. Windows normally wants to be on the first partition with the MBR, the master boot record. You would specify this partition. At the prompt enter

```
grub-install /dev/sda1
```

This will re-install your current GRUB boot loader, assuming that Windows is included in the GRUB configuration. You can then reboot, and the GRUB boot loader will start up. If you are adding Windows for the first time, you will have to add an entry for it in the **/boot/grub/grub.conf** file to have it accessible from the boot loader.

Tip: If even your Linux rescue discs are unable to access your system, you could use a Fedora Live CD to start up Fedora, and then manually mount your Fedora partitions. You will need to know your partition device names (use GParted). Once mounted, you can access the system files on the mounted partition and make any needed changes.

FirstAidKit

Fedora has introduced FirstAidKit which is designed to automatically detect and fix certain common problems. FirstAidKit is still under development. You will need to install FirstAidKit and its plugins. There are plugins for different problem areas, such as an xserver plugin for display problems, or passwd for password file corruption, and mdadm_conf for RAID file system issues. Currently under development are bootloader, partitions, RPM, and file system plugins. Use the **--list** option to list available plugins.

```
firstaidkit --list
```

See the following sites for more information.

```
https://fedorahosted.org/firstaidkit/
http://fedoraproject.org/wiki/Features/FirstAidKit
```

Plugins are different recovery processes, such as a recover process for the password file or a recover process for the Xserver (display). Within each recover process you can have certain recover task, like resetting the root password or using a different Xserver driver. FirstAidKit also support flows. These are ways to structure certain tasks in a recover process. You can think of them as different kinds of tasks to perform. The passwd plugin (recovery process) supports a **resetRoot** flow, to reset the root password. The xserver plugin supports a **diagnose**, **fix**, and **force** flow to just diagnose problems, fix them, or to force a fix. The RAID plugin supports only a **diagnose** and **fix** flow. Figure 2-12 shows the flow for the current plugins. To find out the flows for a specific plugin, as well as other information about the plugin, you can use the **firstaidkit** command with the **--info** option. The following displays detailed information about the xserver plugin.

```
firstaidkit --info xserver
```

If you want to use FirstAidKit in a rescue mode, first start up the Rescue mode from the Install DVD as described in the previous section, Rescue. Then you can use the **firstaidkit** command with the **-a** option to perform a general diagnostic.

```
firstaidkit -a
```

To fix a problem add the **fix** option.

```
firstaidkit -a fix
```

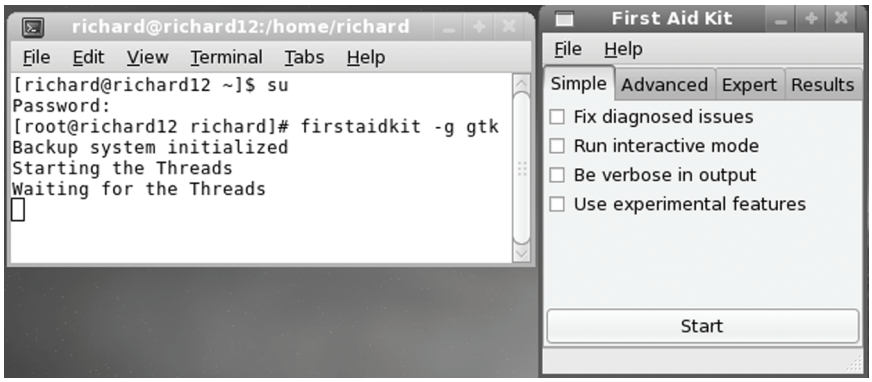


Figure 2-11: FirstAidKit GTK interface

To perform a flow on a particular plugin use the -f option with the plugin and the flow specified. The following will perform a diagnostic on the xserver.

```
firstaidkit -f xserver diagnose
```

This command will reset the root user password. It will generate a random password. Be sure to copy it down, or change your root user password immediately to something you can remember.

```
firstaidkit -f passwd resetRoot
```

Also available is the FirstAidKit GNOME interface which you can run with the following command. Login as the root user first with the **su** command in a terminal window.

```
firstaidkit -g -gtk
```

The FirstAidKit GTK interface will open displaying four tabs: Simple, Advanced, Expert, and Results (see Figure 2-11). The Simple screen lets you run a diagnostic with supported plugins. You can set options like fixing issues or display detailed results.

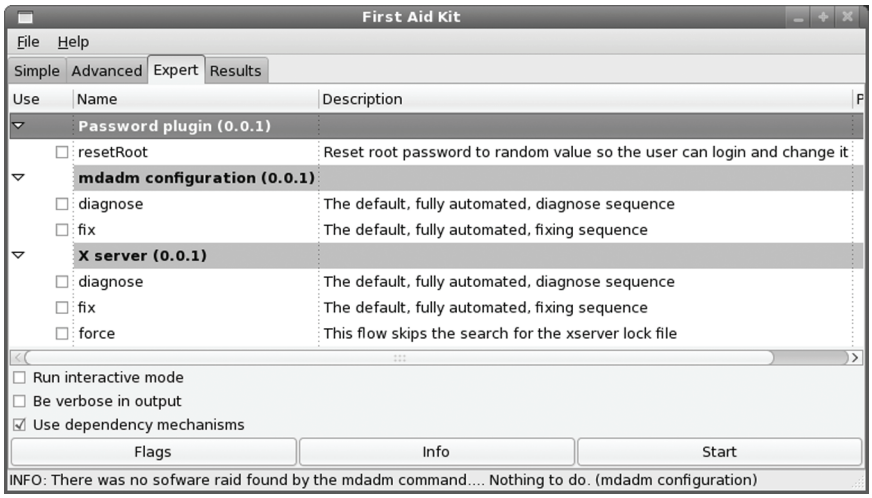


Figure 2-12: FirstAidKit GTK interface Expert tab

The Advanced tab has a drop-down menu that list different flows. The currently supported ones are listed: diagnose, fix. force, and resetRoot. You can only pick one. The Expert tab will let you choose several for specific plugins. It also lists several options: interactive, verbose output, experimental features, and dependency mechanism. The dependency mechanism is on by default and checks for dependencies between plugins.

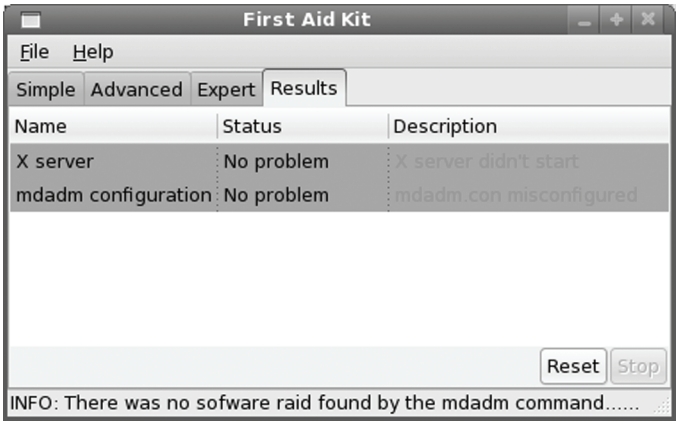


Figure 2-13: FirstAidKit GTK interface Results tab

The Expert tab lets you specify flows by plugin, and the Results tab display results of the diagnostic and fix operations (see Figures 2-12 and 2-13).

Cobbler Install Server

Cobbler is an install server that lets you automatically install and update numerous systems, including virtual systems. It also supports DHCP, DNS, and YUM mirror configuration. Configuration is held in the **/etc/cobbler/conf.d** directory.

You can find out more about Cobbler at its Web site:

```
https://fedorahosted.org/cobbler/
```

For documentation on your system you can use the following URL in any Web browser.

```
http://localhost/cobbler/webui/clobber.html
```

Also check the man pages for **cobbler** and **koan**. **koan** is the Kickstart over a network, designed for use with cobbler.

Cobbler also provides a Web interface for managing your installs. CobblerWebInterface. See the following for details:

```
https://fedorahosted.org/cobbler/wiki/CobblerWebInterface
```

In a Web browser enter:

```
http://localhost/cobbler/webui/webui.html
```

Authentication and access is managed by module specifications in the **/etc/httpd/conf.d/cobbler.conf** file. By default access is denied to all users, though you can specify authentication using Web passwords (**htpasswd**), LDAP, and Kerberos.

fedora

3. System Startup and Services

Upstart

Upstart and Runlevels: event.d and init.d

System Startup files and scripts

System Runlevels: telinit, inittab, and shutdown

Managing Services

System start up is now managed by the Upstart service. The original System V init format for start individual services is still in place, but now connected to Upstart, which performs the actual work. A single Linux system can provide several different kinds of services, ranging from security to administration and including more obvious Internet services like web and FTP sites, e-mail, and printing. Security tools such as SSH and Kerberos run as services, along with administrative network tools such as DHCP and LDAP. The network connection interface is itself a service that you can restart at will. Each service operates as a continually running daemon looking for requests for its particular services.

Upstart

Linux systems traditionally used the Unix System V init daemon to manage services by setting up runlevels at which they could be started or shutdown. Fedora has since replaced the SystemV init deamon with the Upstart init daemon, while maintaining the System V init runlevel structure for compatibility purposes. Whereas the System V init daemon would start certain services when the entire system started up or shutdown, Upstart is entirely event driven. When an event occurs invoking the need for a service, then the service is started. This event oriented approach is designed to work well with removable devices. When a device is added or removed, this change becomes an event that the Upstart daemon detects and then runs any appropriate associated scripts. System V init daemon only ran scripts when its runlevels changed. It saw only runlevels, not events.

Structurally, Upstart can detect and respond to any event. Eventually it may implement scheduled events, replacing cron, atd, an anacron schedulers, as well as support for on demand service now managed by xinetd. You can find out more about Upstart at:

```
http://fedoraproject.org/wiki/Features/Upstart
http://upstart.ubuntu.com
http://upstart.ubuntu.com/wiki/
```

Also check the **events** man page. Upstart also has its own versions of **shutdown**, **reboot**, **halt**, **init**, and **telinit**, each with their own man page. These versions are used on Fedora, instead of the original Linux versions used in previous releases.

Upstart will detect events, and run scripts in the **/etc/event.d** directory for those events. These scripts define jobs that Upstart can then run. Jobs that can be performed by Upstart are defined in scripts in Upstart event directory. Here you will finds Upstart job scripts for emulating runlevels like the rc2 script, as well as system services like TTY terminal connections. In effect, Upstart jobs replace the entries that used to be in the SysVinit's **/etc/inittab** file.

Upstart operates by running jobs. These jobs are defined in job definition files located in the **/etc/event.d** directory. Jobs are already defined for System V init runlevel emulation, TTY system services, and for certain tasks like Ctrl-Alt-Delete event to restart your system. A job script will specify an event, the action to take for that event, and any commands to run for that event. The commands can be either a single command run by an exec operation, or a set of command encased in **script** and **end script** stanza.

To have a job started automatically when a certain event occurs you place a **start on** directive in its job file, specifying the event. You can use a **stop on** directive to stop the event automatically. You can have several start on directives, each for different events. The **start on** directive for the control-alt-delete job is shown here.

```
start on control-alt-delete
```

In the control-alt-delete job, you have a control-alt-delete event that runs the shutdown command with the **-r** option to restart. The **exec** command is used to run a shell command directly.

```
start on control-alt-delete
     exec /sbin/shutdown -r -now
```

Many jobs defined both a **start on** and **stop on** directive, for starting and stopping the job. The event can also take an argument. In the **rc5** job definition which controls runlevel 5 emulation has **start** and **stop** events. The **rc5** job will start when a runlevel event occurs with the argument 5. It will stop when any runlevel even occurs that has a number other than 5, **runlevel** without an argument

```
start on runlevel 5
stop on runlevel
```

Alternatively you can use the brackets with the not operator, **!**, as in **[!5].** This command is used in the **prefdm**, preferred display manager, startup script, which is started after **rc5** job is stops.

```
start on stopped rc5
```

The startup script will end if the system is not starting up in runlevel 5.

```
stop on runlevel [!5]
```

If it is in runlevel 5, then script continues on to run the actual X11 **prefdm** script.

```
exec /etc/X11/prefdm
```

A **stopped** or **started** event indicates when some other job has been started or stopped. The following start on directive would start its job whenever the **myjob** job started.

```
start on started myjob
```

A job can also refer to a script, as in the runlevel scripts **rc1** through **rc5**. To run a script after another has finished, you refer to the script name rather than the runlevel. The following will start the script after the **rc2** script has finished.

```
start on stopped rc2
```

Note: An Upstart job definition script does not need to have a **`start on`** directive. It could be started manually with the **`start`** command.

The **startup** event indicates system startup. The **rcS** job (single user mode) will be started up initially whenever you system starts up. In its **/etc/event.d/rcS** job definition file you will find the following:

```
start on startup
```

The TTY1 job used for terminal services will have several start and stop directives, automatically starting at runlevel 2, 3, 4, and 5 (**prefdm**) events, and stopping on runlevel 0, 1, and 6 events. For runlevels 2, 3, and 4, the TTY1 job is started after their **rc** runlevel scripts end, **rc2**, **rc3**, and **rc4**.

```
start on stopped rc2
start on stopped rc3
```

```
start on stopped rc4
start on started prefdm

stop on runlevel 0
stop on runlevel 1
stop on runlevel 6

respawn
exec /sbin/getty tty1
```

To run several command you encase the command in a **script** stanza. A script stanza begins with the **script** keyword and ends with **end script**. Most complex jobs use a script stanza. The **rc5** job script shown here will start on a runlevel event with the 5 argument. Then the commands in the script structure are run, setting the runlevel to 5 with the **runlevel** command. Then the **/etc/init.d/rc** script is run with the 5 argument to start up any **/etc/init.d** service scripts for runlevel 5, emulating a runlevel change to 5.

/etc/event.d/rc5

```
start on runlevel 5
stop on runlevel

console output
script
       set $(runlevel --set 5 || true)
       if [ "$1" != "unknown" ]; then
           PREVLEVEL=$1
           RUNLEVEL=$2
           export PREVLEVEL RUNLEVEL
       fi

       exec /etc/init.d/rc 5
end script
```

The **rc5** job is started when **telinit** in the **rcS** script triggers a runlevel event with a 5 argument (the **rcS** script is always run first). This makes the start on command at the top of the **rc5** script true.

```
start on runlevel 5
```

For any runlevel events that are not 5, the **rc5** job will be stopped.

```
stop on runlevel
```

The **console** directive specifies where the job output goes, usually to the console (**output**).

```
console output
```

In the **rcS** script, which is always run first, Fedora uses some complicated scripting to determine the upstart runlevel script to start. First a startup event is confirmed using the **UPSTART_EVENT** variable.

```
if [ "$UPSTART_EVENT" == "startup" ]; then
```

A listing of the **/etc/event.d/rcS** scrip is shown here.

/etc/event.d/rcS

```
# rcS - runlevel compatibility
#
# This task runs the old sysv-rc startup scripts.

start on startup
stop on runlevel

# Note: there can be no previous runlevel here, if we have one it's bad
# information (we enter rc1 not rcS for maintenance). Run /etc/rc.d/rc
# without information so that it defaults to previous=N runlevel=S.
console output
script
       runlevel --set S >/dev/null || true
       /etc/rc.d/rc.sysinit
       runlevel --reboot || true
end script

post-stop script
       if [ "$UPSTART_EVENT" == "startup" ]; then
              runlevel=$(/bin/awk -F ':' '$3 == "initdefault" { print $2 }'
/etc/inittab)
              [ -z "$runlevel" ] && runlevel="3"
              for t in $(cat /proc/cmdline); do
                     case $t in
                            -s|single|S|s) runlevel="S" ;;
                            [1-9])       runlevel="$t" ;;
                     esac
              done
              exec telinit $runlevel
       fi
end script
```

The **/etc/inittab** file is read and the default runlevel determined from its **initdefault** entry.

```
runlevel=$(/bin/awk -F ':' '$3 == "initdefault" { print $2 }' /etc/inittab)
```

If there is no default runlevel, the runlevel 3 is set as the default.

```
[ -z "$runlevel" ] && runlevel="3"
```

If a runlevel number was entered on the command line, or the GRUB command kernel line, that runelvel is used. This allows you to select the runlevel you wan to start from the GRUB entry for a particular kernel. Just add then runlevel number at the end (as is possible in previous Fedora releases). For the Single user mode (recovery), S, you can enter **s**, **S**, or **single**.

```
for t in $(cat /proc/cmdline); do
       case $t in
              s|single|S|s) runlevel="S" ;;
               [1-9])       runlevel="$t" ;;
       esac
done
```

Once the runlevel is determined, it is started with a **telinit** command that changes to that runlevel, which, in turn, runs the associated Upstart runlevel script, like **/etc/event.d/rc5** for runlevel 5.

```
exec telinit $runlevel
```

You can think of the Upstart daemon managing a set of jobs as similar to how the shell manages background jobs. With the **start** and **stop** commands you can start and stop any job. Use status to find out the **status** of a job. You can either login as the **root** user or use the **sudo** command to run these commands.

```
$ stop tty1
tty1 (stop) running, process 5109
tty1 (stop) pre-stop, (main) process 5109
tty1 (stop) stopping, process 5109
tty1 (stop) killed, process 5109
tty1 (stop) post-stop
tty1 (stop) waiting
$ start tty1
tty1 (start) waiting
tty1 (start) starting
tty1 (start) pre-start
tty1 (start) spawned, process 8457
tty1 (start) post-start, (main) process 8457
tty1 (start) running, process 8457
$ status tty1
tty1 (start) running, process 8457
```

The **initctl** command with the **list** option will display a complete list of current Upstart jobs. You can add a pattern to search or a particular job. The **initctl** command also has **start**, **stop**, and **status** options form managing jobs.

```
initctl list
```

You could also use the **emit** command to manually trigger an event that would run a certain job.

Note: An Upstart job definition script does not have to have a **start on** directive. I could just be started manually with the **start** command.

Upstart and Runlevels: event.d and init.d

Fedora still maintains SysVinit startup and shutdown scripts in the **/etc/init.d** directory. Upstart will invoke the **/etc/rc.d/rc** script with a runlevel number, which in turn will run the associated service scripts in the **/etc/init.d** directory. You can also run these scripts directly to start and stop a service. In effect, the supporting structure for runlevels remains the same, though runlevels are now invoked by Upstart.

You can start up your system at different levels with certain capabilities. For example, you can run your system at an administrative level, locking out user access. Normal full operations are activated by simply running your system at a certain level of operational capability such as supporting multi-user access or graphical interfaces. These levels (also known as states or modes) are referred to as *runlevels,* the level of support that you are running your system at.

Note: You can select certain services to run and the runlevel at which to run them. Most services are servers like a web server or proxy server. Other services provide security, such as SSH or Kerberos. On Fedora you can **system-config-services** to turn on and off services, specifying the runlevel. The default is runlevel 5.

Runlevels

Traditionally, a Linux system has several runlevels, numbered from 0 to 6. Support for these is now emulated by Upstart. When you power up your system, you enter the default runlevel. Runlevels 0, 1, and 6 are special runlevels that perform specific functions. Runlevel 0 is the power-down state and is invoked by the **`halt`** command to shut down the system. Runlevel 6 is the reboot state—it shuts down the system and reboots. Runlevel 1 is the single-user state, which allows access only to the superuser and does not run any network services. This enables you, as the administrator, to perform administrative actions without interference from others.

Other runlevels reflect how you want the system to be used. Fedora uses runlevel 5 for graphical logins and runlevel 3 for the command line interface only.

Tip: You can use the single-user runlevel (1) as a recovery mode state, allowing you to start up your system without running startup scripts for services like DNS. This is helpful if your system hangs when you try to start such services. Networking is disabled, as well as any multi-user access. You can also use **`linux -s`** at the boot prompt to enter runlevel 1. If you want to enter the single-user state and also run the startup scripts, you can use the special **`s`** or **`S`** runlevel.

Runlevels in event.d directory

Technically, runlevels are no longer implemented. Upstart service does not use actual runlevels, it emulates them. To maintain compatibility with System V compliant Linux and Unix applications, Upstart does provide a runlevel compatibility scripts in the **/etc/event.d** directory that emulate System V init. To start up a runlevel, it uses **telinit** with the runlevel number to start. When your system starts up, it uses the default runlevel as specified by the **rcS** script in the **/etc/event.d** directory. There are runlevel scripts for each runlevel in the **/etc/event.d** directory. The default runlevel is 5, which will run the **/etc/event.d/rc5** script, which in turn invokes **telinit** with the argument **5**. The **telinit** script, along with other runlevel scripts like **halt**, **reboot**, and **runlevel**, are actually Upstart versions, included in the **upstart** software package.

default runlevel

The default runlevel is 5 as initially setup in your **/etc/inittab** file. You can change this default runlevel, to 2, 3, or 4 if you wish. To actually change the default runlevel safely, you can create an **/etc/inittab** file and place and **initdefault** entry in it (use **vi** or **emacs** if editing from a command line interface, or **gedit** from the desktop). You will have to edit the file with administrative access using the **sudo** command.

```
sudo gedit /etc/inittab
```

This file is just a dummy file used only by **rcS** to read the default entry. A sample default entry is shown here, changing the default to runelvel 3. Your **/etc/inittab** file would have only this line.

```
id:3:initdefault:
```

On Fedora, if the **/etc/inittab** file is missing or corrupted, then the default runlevel is 3, the command line interface. Your system will start up automatically in runlevel 3 using just the command line interface. To restore 5 as the default runlevel (desktop), you would have to create the **/etc/inittab** file with just a following **initdefault** line.

```
id:5:initdefault:
```

Tip: If you used **/etc/inittab** to change the default runlevel, and you want to change quickly to using runlevel 3 as the default without having to edit the **/etc/inittab** file, you can just remove the **/etc/inittab** file.

System Startup files and scripts

Each time you start your system, the Upstart init daemon starts up runlevels defined in startup scripts. Currently most services still use the older System V init method for starting up services using runlevels. Upstart will use its event based init daemon to emulate the System V init structure, allowing many services to run as if they were using the older System V init daemon. Eventually, service applications will be rewritten to use Upstart directly, without the need for System V init emulation.

The System V startup procedure runs a series of startup script from system service scripts located in your **/etc/init.d** directory. It uses links in directories with the name **rc***N***.d**, to determine what service scripts to run. The *N* in the name is a number from 1 to 6 indicating a runlevel, like **rc5.d** for runlevel 5. These initialization files are organized according to different tasks. You should not have to change any of these files (see Table 3-1).

File	Description
/etc/event.d	Upstart job files that actually start services and processes.
/etc/rc*N***.d**	Directories that holds system startup and shutdown files, where *N* is the runlevel. The directories hold links to scripts in the **/etc/init.d** directory.
/etc/rc.local	Initialization file for your own commands; you can freely edit this file to add your own startup commands; this is the last startup file executed.
/etc/init.d	Directory that holds system service scripts.
/etc/init.d/rc	Runlevel emulation by Upstart. Upstart runs this script to emulate runlevel changes.

Table 3-1: System Startup Files and Directories

rc.local

The **/etc/rc.local** file is the last initialization file executed. You can place commands of your own here. When you shut down your system, the system calls the **halt** file, which contains shutdown commands. The files in **init.d** are then called to shut down daemons, and the file systems are unmounted. **halt** is located in the **init.d** directory.

/etc/init.d

The **/etc/init.d** directory is designed primarily to hold scripts that start up and shut down different specialized daemons, such as network and printer daemons and those for font and web servers. These files perform double duty, starting a daemon when the system starts up and shutting down the daemon when the system shuts down. The files in **init.d** are designed in a way to make it easy to write scripts for starting up and shutting down specialized applications. Many of these files are set up for you automatically. You shouldn't need to change them. If you do change them, be sure you know how these files work first.

When your system starts up, several programs are automatically started and run continuously to provide services, such as a website or print servers. Depending on what kind of services you want your system to provide, you can add or remove items in a list of services to be started automatically. For example, the web server is run automatically when your system starts up. If you are not hosting a website, you have no need for the web server. You can prevent the service from starting, removing an extra task the system does not need to perform, freeing up resources, and possibly reducing potential security holes. Several of the servers and daemons perform necessary tasks. The **sendmail** server enables you to send messages across networks, and the **cupsd** server performs printing operations.

/etc/init.d/rc

The **/etc/init.d/rc** script is used by Upstart to emulate System V runlevel changes. The script takes as its argument a runlevel. It then checks the **/etc/rc***N***.d** links for the runlevel to determine what services to start or stop.

/etc/event.d

In the **/etc/event.d** directory, Upstart maintains event scripts for different runlevels, **rc***N*. For runlevel 5, there is an **rc5** script. This script simply runs the **/etc/init.d/rc** script with the number 5 as its argument. The **/etc/init.d/rc** script will then search the **/etc/rc5.d** directory for links specifying which service script to start and which to stop.

When your system starts, the **/etc/event.d/rcS** script is run which will start up with runlevel 3 or the runlevel specified in an **/etc/inittab** dummy file. The **/etc/inittab** file will be set up during installation with runlevel 5, making runlevel 5 the default. For runlevel 5, the **/etc/event.d/rc5** script will invoke the **/etc/init.d/rc** script the argument 5.

Note: Keep in mind that the System V runlevel system does not exist on Fedora though its service management structure does. Instead, Upstart emulates System V runlevels using the **/etc/init.d/rc** script and the same service links and startup scripts in the **/etc/init.d** and **/etc/rc***N***.d** directories.

/etc/rc.d/init.d: System V Init Scripts

You can manage the startup and shutdown of server daemons with special service scripts located in the **/etc/rc.d/init.d** directory. These scripts often have the same name as the service's program. For example, for the **/usr/sbin/httpd** Web server program, the corresponding script is called **`/etc/rc.d/init.d/httpd`**. This script starts and stops the Web server. This method of using **init.d** service scripts to start servers is called *SysV Init,* after the method used in Unix System V. Some of the more commonly used service scripts are listed in Table 3-2.

Service Script	Description
network	Operations to start up or shut down your network connections.
xinetd	Operations to start up or shut down the **xinetd** daemon.
autofs	Automatic file system mounting
cups	The CUPS printer daemon
cpuspeed	Service to manage CPU speed
dhcpd	Dynamic Host Configuration Protocol daemon
httpd	Apache Web server
innd	Internet News service
ipsec	IPsec secure VPN service
iptables	Controls the IPtables daemon
ip6tables	IPtables for IP protocol version 6
krb5kdc	Kerberos kdc server
kudzu	Detects new hardware
ldap	LDAP service
nfs	Network Filesystem
postfix	Postfix mail server
sendmail	The Sendmail MTA daemon
smb	Samba for Windows hosts
squid	Squid proxy-cache server
sshd	Secure Shell daemon
syslog	System logging daemon
vsftpd	Very Secure FTP server
xfs	X Window System font server
ypbind	Network Information Service (NIS)

Table 3-3: Collection of Service Scripts in /etc/rc.d/init.d

The service scripts in the **/etc/rc.d/init.d** directory can be executed automatically whenever you boot your system. Be careful when accessing these scripts, however. These start essential programs, such as your network interface and your printer daemon. These init scripts are accessed from links in subdirectories set up for each possible runlevel. The **/etc/rc.d** directory holds a set of subdirectories whose names have the format **rc*n*.d**, where *n* is a number referring to a runlevel (there are also links in the **/etc** directory directly to the **/etc/rc.d** runlevel subdirectories). The **rc** script detects the runlevel in which the system was started, and then executes only the service scripts specified in the subdirectory for that runlevel. When you start your system, the **rc** script executes the service scripts specified in the **rc3.d** directory, if you are performing a command

line login, or the **rc5.d** directory, if you are using a graphical login. The **rc3.d** and **rc5.d** directories hold symbolic links to certain service scripts in the **/etc/rc.d/init.d** directory. Thus, the **httpd** script in the **/etc/rc.d/init.d** directory is actually called through a symbolic link in the **rc3.d** or the **rc5.d** directory. The symbolic link for the **/etc/rc.d/httpd** script in the **rc3.d** directory is **S85httpd**. The *S* prefixing the link stands for "startup"; thus, the link calls the corresponding **init.d** script with the **start** option. The number indicates the order in which service scripts are run; lower numbers run first. **S85httpd** invokes **/etc/rc.d/init.d/httpd** with the option **start**. If you change the name of the link to start with a *K,* the script is invoked with the **stop** option, stopping it. Such links are used in the runlevels 0 and 6 directories, **rc6.d** and **rc0.d**. Runlevel 0 halts the system, and runlevel 6 reboots it. You can use the **runlevel** command to find out what runlevel you are currently operating at.

System Runlevels: telinit, inittab, and shutdown

A Linux system can run in different levels, depending on the capabilities you want to give it. For example, you can run your system at an administrative level, locking out user access. Normal full operations are activated by simply running your system at a certain level of operational capability such as supporting multi-user access or graphical interfaces. These levels (also known as states or modes) are referred to as *runlevels,* the level of support that you are running your system at.

Runlevels

A Linux system has several runlevels, numbered from 0 to 6. When you power up your system, you enter the default runlevel. Runlevels 0, 1, and 6 are special runlevels that perform specific functions. Runlevel 0 is the power-down state and is invoked by the **halt** command to shut down the system. Runlevel 6 is the reboot state—it shuts down the system and reboots. Runlevel 1 is the single-user state, which allows access only to the superuser and does not run any network services. This enables you, as the administrator, to perform administrative actions without interference from others.

Other runlevels reflect how you want the system to be used. Runlevel 2 is a partial multi-user state, allowing access by multiple users, but without network services like NFS or **xinetd** (eXtended InterNET services daemon). This level is useful for a system that is not part of a network. Both runlevel 3 and runlevel 5 run a fully operational Linux system, with multi-user support and remote file sharing access. They differ in terms of the interface they use. Runlevel 3 starts up your system with the command line interface (also known as the text mode interface). Runlevel 5 starts up your system with an X session, running the X Window System server and invoking a graphical login, using display managers, such as gdm or xdm. If you choose to use graphical logins during installation, runlevel 5 will be your default runlevel. Linux provide two keyboard sequences to let you switch between the two during a login session: CTRL-ALT-F6 changes from the graphical interface (runlevel 5) to the command line interface (runlevel 3), and CTRL-ALT-F1 changes from the command line interface to the graphical interface. The runlevels are listed in Table 3-3.

Changing runlevels can be helpful if you have problems at a particular runlevel. For example, if your video card is not installed properly, then any attempt to start up in runlevel 5 will likely fail, as this level immediately starts your graphical interface. Instead, you should use the command line interface, runlevel 3, to fix your video card installation.

Tip: You can use the single-user runlevel (1) as a recovery mode state, allowing you to start up your system without running startup scripts for services like DNS. This is helpful if your system hangs when you try to start such services. Networking is disabled, as well as any multi-user access. You can also use **`linux -s`** at the boot prompt to enter runlevel 1. If you want to enter the single-user state and also run the startup scripts, you can use the special **`s`** or **`S`** runlevels.

System Runlevels (states)	Description
0	Halt (do *not* set the default to this level); shuts down the system completely.
1	Administrative single-user mode; denies other users access to the system, but allows root access to the entire multi-user file system. Startup scripts are not run. (Use **`s`** or **`S`** to enter single-user mode with startup scripts run).
2	Multi-user, without network services like NFS, **xinetd**, and NIS (the same as 3, but you do not have networking).
3	Full multi-user mode with login to command line interface; allows remote file sharing with other systems on your network. Also referred to as the *text mode state*.
4	Unused.
5	Full multi-user mode that starts up in an X session, initiating a graphical login; allows remote file sharing with other systems on your network (same as 3, but with graphical login).
6	Reboots; shuts down and restarts the system (do *not* set the default to this).

Table 3-3: System Runlevels (States)

Runlevels in initab

When your system starts up, it uses the default runlevel as specified in the default **`init`** entry in the **/etc/inittab** file. For example, if your default init runlevel is 5 (the graphical login), the default **`init`** entry in the **/etc/inittab** file would be

```
id:5:initdefault:
```

The **inittab** file performs no other function. It is merely a storage file for the default runlevel, which is read by Upstart.

You can change the default runlevel by editing the **/etc/inittab** file and changing the **`initdefault`** entry. As an example, if the default runlevel is 3 (command line), the entry for your default runlevel in the **/etc/inittab** file should look like the following:

```
id:3:initdefault:
```

You can change the 3 to a 5, to change your default runlevel from the command line interface (3) to the graphical login (5). Change only this number and nothing else.

```
id:5:initdefault:
```

Tip: If your **/etc/inittab** file becomes corrupted, you will automatically boot to runlevel 3, the command line interface. You can then edit or replace the **/etc/inittab** file to reset your default runlevel.

Changing Runlevels with telinit

No matter what runlevel you start in, you can change from one runlevel to another with the **telinit** command. If your default runlevel is 3, you power up in runlevel 3, but you can change to, say, runlevel 5 with **telinit 5**. The command **telinit 0** shuts down your system. In the next example, the **telinit** command changes to runlevel 1, the administrative state:

```
# telinit 1
```

Before Upstart was implemented, you could also use **init** to also change runlevels. With Upstart, both **telinit** and **init** are now Upstart version of the original Unix commands. The **telinit** command is always used to change runlevels. If you use init with a runlevel number, it now merely invokes **telinit** to make the change. The Upstart **init** command is designed to only perform the actual startup operations and is automatically invoked when your system starts up. It will run the scripts in the **/etc/event.d** directory on start up.

The runlevel Command

Use the **runlevel** command to see what state you are currently running in. It lists the previous state followed by the current one. If you have not changed states, the previous state will be listed as N, indicating no previous state. This is the case for the state you boot up in. In the next example, the system is running in state 3, with no previous state change:

```
# runlevel
N 3
```

Shutdown

Although you can power down the system with the **telinit** command and the 0 state, you can also use the **shutdown** command. The **shutdown** command has a time argument that gives users on the system a warning before you power down. You can specify an exact time to shut down, or a period of minutes from the current time. The exact time is specified by *hh***:***mm* for the hour and minutes. The period of time is indicated by a **+** and the number of minutes. Keep in mind that the shutdown command used is the Upstart version, which will use Upstart to actually shutdown the system.

. The **shutdown** command takes several options with which you can specify how you want your system shut down. The **-h** option, which stands for halt, simply shuts down the system, whereas the **-r** option shuts down the system and then reboots it. In the next example, the system is shut down after ten minutes:

```
# shutdown -h +10
```

To shut down the system immediately, you can use **+0** or the word **now**. The shutdown options are listed in Table 3-4. The following example shuts down the system immediately and then reboots:

```
# shutdown -r now
```

Command	Description
shutdown [-rkhncft] *time* **[***warning***]**	Shuts the system down after the specified time period, issuing warnings to users; you can specify a warning message of your own after the time argument; if neither **-h** nor **-r** is specified to shut down the system, the system sets to the administrative mode, runlevel state 1.
Argument	
Time	Has two possible formats: it can be an absolute time in the format *hh*:*mm,* with *hh* as the hour (one or two digits) and *mm* as the minute (in two digits); it can also be in the format +*m,* with *m* as the number of minutes to wait; the word **now** is an alias for **+0**.
Option	
-t *sec*	Tells **init** to wait *sec* seconds between sending processes the warning and the kill signals, before changing to another runlevel.
-k	Doesn't actually shut down; only sends the warning messages to everybody.
-r	Reboots after shutdown, runlevel state 6.
-h	Halts after shutdown, runlevel state 0.
-n	Doesn't call **init** to do the shutdown; you do it yourself.
-f	Skips file system checking (fsck) on reboot.
-c	Cancels an already running shutdown; no time argument.

Table 3-4: System Shutdown Options

With the **shutdown** command, you can include a warning message to be sent to all users currently logged in, giving them time to finish what they are doing before you shut them down.

```
# shutdown -h +5 "System needs a rest"
```

If you do not specify either the **-h** or the **-r** options, the **shutdown** command shuts down the multi-user mode and shifts you to an administrative single-user mode. In effect, your system state changes from 3 (multi-user state) to 1 (administrative single-user state). Only the root user is active, allowing the root user to perform any necessary system administrative operations with which other users might interfere.

Tip: You can also shut down your system from the GNOME or KDE desktops.

Managing Services

You can select certain services to run and the runlevel at which to run them. Most services are servers like a Web server or proxy server. Other services provide security, such as SSH or Kerberos. You can decide which services to use with the **chkconfig**, **service**, or system-config-services tools.

system-config-services

The system-config-services tool provides an interface displaying a simple list of services from which you can select the ones you want to start up (System | Administration | Services) and have started automatically. On the system-config-services tool, the side pane displays a listing of installed daemons and servers (see Figure 3-1). A status ions indicate whether a service is enabled or not (green or red), and if they it is running. The right pane will display the current status of a selected service and its description.

Several services are already selected by default, like **network** which runs your network connections and **haldaemon** which detects your hardware. Others like Wine which runs the Windows emulator, and **smb** which runs Samba Windows network support, are optional. You can use system-config-services to run them.

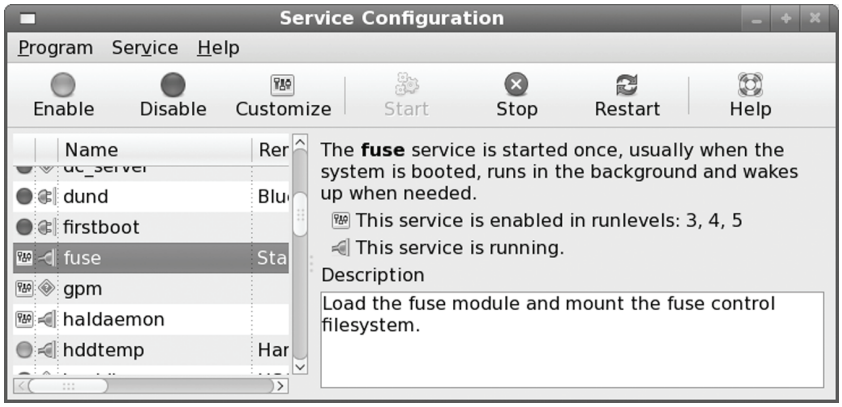


Figure 3-1: Services: system-config-services

Enabling a service will have it start up automatically every time your system boots up. Once enabled, you can then manually start and stop the service if you wish. To enable a service, select its entry and click the Enable button on the Toolbar. You will see the status icon turn green. You can then click the Start button to start the service manually, and then Stop to stop it.

Figure 3-2: Service runlevel customization

To have a service start automatically, it has to be configured to run at standard runlevels, at least runlevel 5 and 3. Most services are configured to run at runlevels 2, 3, 4, and 5, when you enable them. Some are not. You can customize service to run at different runlevels. Select the service's entry and then click the Customize button. This opens a window listing all the runlevels with check boxes for each (see Figure 3-2). Click the runlevels you want the service to start at.

chkconfig

To configure a service to start up automatically, you can use the system-config-services tool available on the desktop or the **chkconfig** tool, which is run on a command line. The system-config-services tool will display a list of available services, letting you choose the ones you want to start and deselect. The **chkconfig** command uses the **on** and **off** options to select and deselect services for startup.

```
/sbin/chkconfig nmb on
```

You can specify the service you want to start and the level you want to start it at with the **chkconfig** command. Unlike other service management tools, **chkconfig** works equally well on stand-alone and **xinetd** services. Though stand-alone services can be run at any runlevel, you can also turn **xinetd** services on or off for the runlevels that **xinetd** runs in. Table 3-5 lists the different **chkconfig** options.

Listing Services with chkconfig

To see a list of services, use the **--list** option. A sampling of services managed by **chkconfig** is shown here. The on or off status of the service is shown at each runlevel, as are **xinetd** services and their statuses:

```
chkconfig -list
dhcpd  0:off 1:off 2:off 3:off 4:off 5:off 6:off
httpd  0:off 1:off 2:off 3:off 4:off 5:off 6:off
named  0:off 1:off 2:off 3:off 4:off 5:off 6:off
lpd    0:off 1:off 2:on  3:on  4:on  5:on  6:off
nfs    0:off 1:off 2:off 3:off 4:off 5:off 6:off
crond  0:off 1:off 2:on  3:on  4:on  5:on  6:off
xinetd 0:off 1:off 2:off 3:on  4:on  5:on  6:off
xinetd based services:
     time:       off
     finger:     off
     pop3s:      off
     swat:       on
```

Starting and Stopping Services with chkconfig

You use the **on** option to have a service started at specified runlevels, and the **off** option to disable it. You can specify the runlevel to affect with the **--level** option. If no level is specified, **chkconfig** will use any **chkconfig** default information in a service's **init.d** service script. Red Hat and Fedora install their services with **chkconfig** default information already entered (if this is missing, **chkconfig** will use runlevels 3, 4, and 5). The following example has the Web server (**httpd**) started at runlevel 5:

```
chkconfig --level 5 httpd on
```

The **off** option configures a service to shut down if the system enters a specified runlevel. The next example shuts down the Web server if runlevel 3 is entered. If the service is not running, it remains shut down:

```
chkconfig --level 3 httpd off
```

The **reset** option restores a service to its **chkconfig** default options as specified in the service's **init.d** service script:

```
chkconfig httpd reset
```

To see just the startup information for a service, you use just the service name with the **--list** option:

```
chkconfig --list httpd
httpd   0:off  1:off 2:off 3:on 4:off 5:on 6:off
```

Enabling and Disabling xinetd Services with chkconfig

The **chkconfig** command can also enable or disable **xinetd** services. Simply enter the **xinetd** service with either an **on** or **off** option. The service will be started up or shut down, and the disable line in its **xinetd** configuration script in the **/etc/xinetd.d** directory will be edited accordingly. For example, to start swat, the Samba configuration server, which runs on **xinetd**, you simply enter

```
chkconfig swat on
chkconfig --list swat
     swat          on
```

Option	Description
--level *runlevel*	Specifies a runlevel to turn on, turn off, or reset a service.
--list *service*	Lists startup information for services at different runlevels. **xinetd** services are just **on** or **off**. With no argument, all services are listed, including **xinetd** services.
--add *service*	Adds a service, creating links in the default specified runlevels (or all runlevels, if none are specified).
--del *service*	Deletes all links for the service (startup and shutdown) in all runlevel directories.
service **on**	Turns a service on, creating a service link in the specified or default runlevel directories.
service **off**	Turns a service off, creating shutdown links in specified or default directories.
service **reset**	Resets service startup information, creating default links as specified in the **chkconfig** entry in the service's **init.d** service script.

Table 3-5: Options for chkconfig

The **swat** configuration file for **xinetd**, **/etc/xinetd.d/swat**, will have its disable line edited to no, as shown here:

```
disable=no
```

If you want to shut down the swat server, you can use the **off** option. This will change the disable line in **/etc/xinetd.d/swat** to read "disable=yes".

```
chkconfig swat off
```

The same procedure works for other **xinetd** services such as the POP3 server and **finger**.

Removing and Adding Services with chkconfig

If you want a service removed entirely from the entire startup and shutdown process in all runlevels, you can use the **--del** option. This removes all startup and shutdown links in all the runlevel directories.

```
chkconfig --del httpd
```

You can also add services to **chkconfig** management with the **--add** option; **chkconfig** will create startup links for the new service in the appropriate startup directories, **/etc/rc.d/rc*n*.d**. If you have previously removed all links for a service, you can restore them with the **add** option.

```
chkconfig --add httpd
```

Configuring xinetd Services for Use by chkconfig

Default runlevel information should be placed in the service scripts that are to be managed by **chkconfig**. Red Hat and Fedora have already placed this information in the service scripts for the services that are installed with its distribution. You can edit these scripts to change the default information if you wish. This information is entered as a line beginning with a **#** sign and followed by the **chkconfig** keyword and a colon. Then you list the default runlevels that the service should start up on, along with the start and stop priorities. The following entry lists runlevels 3 and 5 with a start priority of 85 and a stop of 15. See the section "Service Script Tags" for more information:

```
# chkconfig: 35 85 15
```

Thus when a user turns on the **httpd** service with no level option specified, **chkconfig** will start up **httpd** at runlevels 3 and 5.

```
chkconfig httpd on
```

How chkconfig Works

The **chkconfig** tool works by creating startup and shutdown links in the appropriate runlevel directories in the **/etc/rc.d** directory. For example, when **chkconfig** adds the **httpd** service at runlevel 5, it creates a link in the **/etc/rc.d/rc5.d** directory to the service script **httpd** in the **/etc/rc.d/init.d** directory. When it turns off the Web service from runlevel 3, it creates a shutdown link in the **/etc/rc.d/rc3.d** directory to use the script **httpd** in the **/etc/rc.d/initd** directory to make sure the Web service is not started. In the following example, the user turns on the Web service (**httpd**) on runlevel 3, creating the startup link in **rc5.d**, **S85httpd**, and then turns off the Web service on runlevel 3, creating a shutdown link in **rc3.d**, **K15httpd**.

```
chkconfig --level 5 httpd on
ls /etc/rc.d/rc5.d/*httpd
   /etc/rc.d/rc5.d/S85httpd
chkconfig -level 3 httpd off
ls /etc/rc.d/rc3.d/*httpd
  /etc/rc.d/rc3.d/K15httpd
```

The service Command

To start and stop services manually, you can user either system-config-services or the **service** command. With the **service** command, you list the service with the **stop** argument to stop it, the **start** argument to start it, and the **restart** argument to restart it. The scripts are run from a Terminal window. You will have to first login as the **root** user, using the **su** command, or use the **sudo** command, if configured. The following will start the **smb** Samba service.

```
service smb start
```

You can also use the service's script in the **/etc/init.d** directory, directly. Use the service's script with the **stop** argument to stop it, the **start** argument to start it, and the **restart** argument to restart it. The following will restart the nmb Samba network discovery service

```
/etc/init.d/nmb restart
```

Service Script: /etc/init.d

Most software using RPM packages will automatically install any appropriate service scripts and create the needed links in the appropriate **rc*n*.d** directories, where *n* is the runlevel number. Service scripts, though, can be used for any program you may want run when your system starts up. To have such a program start automatically, you first create a service script for it in the **/etc/rc.d/init.d** directory and then create symbolic links to that script in the **/etc/rc.d/rc3.d** and **/etc/rc.d/rc5.d** directories. A shutdown link (*K*) should also be placed in the **rc6.d** directory used for runlevel 6 (reboot).

Service Script Functions

A simplified version of the service script **httpd** uses on Red Hat and Fedora systems is shown in a later section. You can see the different options, listed in the **/etc/rc.d/init.d/httpd** example, under the **case** statement: **start**, **stop**, **status**, **restart**, and **reload**. If no option is provided (*****), the script use syntax is displayed. The **httpd** script first executes a script to define functions used in these service scripts. The **daemon** function with **httpd** actually executes the **/usr/sbin/httpd** server program.

```
echo -n "Starting httpd: "
 daemon httpd
 echo
 touch /var/lock/subsys/httpd
```

The **killproc** function shuts down the daemon. The lock file and the process ID file (**httpd.pid**) are then deleted:

```
killproc httpd
echo
rm -f /var/lock/subsys/httpd
rm -f /var/run/httpd.pid
```

The **daemon**, **killproc**, and **status** functions are shell scripts defined in the **functions** script, also located in the **init.d** directory. The **functions** script is executed at the beginning of each service script to activate these functions. A list of these functions is provided in Table 3-6.

```
. /etc/rc.d/init.d/functions
```

Init Script Function	Description
daemon [+/-*nicelevel*] *program* [*arguments*] [**&**]	Starts a daemon, if it is not already running.
killproc *program* [*signal*]	Sends a signal to the program; by default it sends a **SIGTERM**, and if the process doesn't stop, it sends a **SIGKILL**. It will also remove any PID files, if it can.
pidofproc *program*	Used by another function, it determines the PID of a program.
status *program*	Displays status information.

Table 3-6: Init Script Functions

Service Script Tags

The beginning of the service script holds tags used to configure the server. These tags, which begin with an initial **#** symbol, are used to provide runtime information about the service to your system. The tags are listed in Table 3-7, along with the service functions. You enter a tag with a preceding **#** symbol, the tag name with a colon, and then the tag arguments. For example, the **processname** tag specifies the name of the program being executed, in this example **httpd**:

```
# processname: httpd
```

If your script starts more than one daemon, you should have a **processname** entry for each. For example, the Samba service starts up both the **smbd** and **nmbd** daemons.

```
# processname: smbd
# processname: nmbd
```

The end of the tag section is indicated by an empty line. After this line, any lines beginning with a **#** are treated as comments. The **chkconfig** line lists the default runlevels that the service should start up on, along with the start and stop priorities. The following entry lists runlevels 3, 4, and 5 with a start priority of 85 and a stop of 15:

```
# chkconfig: 345 85 15
```

For the description, you enter a short explanation of the service, using the \ symbol before a newline to use more than one line.

```
# description: Apache Web server
```

With **config** tags, you specify the configuration files the server may use. In the case of the Apache Web server, there may be three configuration files:

```
# config: /etc/httpd/conf/access.conf
# config: /etc/httpd/conf/httpd.conf
# config: /etc/httpd/conf/srm.conf
```

The **pidfile** entry indicates the file where the server's process ID is held.

Init Script Tags	Description
# chkconfig: *startlevellist startpriority endpriority*	Required. Specifies the default start levels for this service as well as start and end priorities.
# description [*ln***]:** *description of service*	Required. The description of the service, continued with **** characters. Use an initial **#** for any added lines. With the *ln* option, you can specify the language the description is written in.
# autoreload: true	Optional. If this line exists, the daemon checks its configuration files and reloads them automatically when they change.
# processname: *program*	Optional, multiple entries allowed. The name of the program or daemon started in the script.
# config: *configuration-file*	Optional, multiple entries allowed. Specifies a configuration file used by the server.
# pidfile: *pid-file*	Optional, multiple entries allowed. Specifies the PID file.
# probe: true	Optional, used *in place* of **autoreload**, **processname**, **config**, and **pidfile** entries to automatically probe and start the service.

Table 3-7: System V init Script Tags

Installing Service Scripts

The RPM-packaged version for a service includes a service script. For example, an Internet server package includes the service script for that server. Installing the RPM package installs the script in the **/etc/rc.d/init.d** directory and creates its appropriate links in the runlevel directories, such as **/etc/rc.d/rc3.d**. If you decide, instead, to create the server using its source code files, you can then manually install the service script. If no service script exists, you first make a copy of the **httpd** script—renaming it—and then edit the copy to replace all references to **httpd** with the name of the server daemon program. Then place the copy of the script in the **/etc/rc.d/init.d** directory and make a symbolic link to it in the **/etc/rc.d/rc3.d** directory. Or you could use system-config-services to create the link in the **/etc/rc.d/rc3.d** directory. When you start your system now, the new server is automatically started up, running concurrently and waiting for requests.

Service Script Example

As an example, a simplified version of the Web server service script, **/etc.rc.d/init.d/httpd**, is shown here. Most scripts are much more complicated, particularly when determining any arguments or variables a server may need to specify when it starts up. This script has the same name as the Web server daemon, **httpd**:

```
#!/bin/sh
#
# Service script for the Apache Web Server
#
# chkconfig: 35 85 15
# description: Apache is a World Wide Web server. \
# It is used to serve HTML files and CGI.
# processname: httpd
# pidfile: /var/run/httpd.pid
# config: /etc/httpd/conf/access.conf
# config: /etc/httpd/conf/httpd.conf
# config: /etc/httpd/conf/srm.conf
# Source function library.
. /etc/rc.d/init.d/functions

# See how we were called.
case "$1" in
 start)
  echo -n "Starting httpd: "
   daemon httpd
   echo
   touch /var/lock/subsys/httpd
   ;;
  stop)
   killproc httpd
   echo
   rm -f /var/lock/subsys/httpd
   rm -f /var/run/httpd.pid
   ;;
 status)
   status httpd
   ;;
 restart)
   $0 stop
   $0 start
   ;;
 reload)
   echo -n "Reloading httpd: "
   killproc httpd -HUP
   echo
   ;;
 *)
   echo "Usage: $0 {start|stop|restart}"
   exit 1
 esac
exit 0
```

Extended Internet Services Daemon (xinetd)

If your system averages only a few requests for a specific service, you don't need the server for that service running all the time. You need it only when a remote user is accessing its service. The Extended Internet Services Daemon (**xinetd**) manages Internet servers, invoking them only when your system receives a request for their services. **xinetd** checks continuously for any

requests by remote users for a particular Internet service; when it receives a request, it then starts the appropriate server daemon.

The **xinetd** program is designed to be a replacement for **inetd**, providing security enhancements, logging support, and even user notifications. For example, with **xinetd** you can send banner notices to users when they are not able to access a service, telling them why. **xinetd** security capabilities can be used to prevent denial-of-service attacks, limiting remote hosts' simultaneous connections or restricting the rate of incoming connections. **xinetd** also incorporates TCP, providing TCP security without the need to invoke the **tcpd** daemon. Furthermore, you do not have to have a service listed in the **/etc/services** file. **xinetd** can be set up to start any kind of special-purpose server.

Attribute	Description
`ids`	Identifies a service. By default, the service ID is the same as the service name.
`type`	Type of service: `RPC`, `INTERNAL` (provided by **xinetd**), or `UNLISTED` (not listed in a standard system file).
`flags`	Possible flags include `REUSE`, `INTERCEPT`, `NORETRY`, `IDONLY`, `NAMEINARGS` (allows use of `tcpd`), `NODELAY`, and `DISABLE` (disable the service). See the **xinetd.conf** Man page for more details.
`disable`	Specify `yes` to disable the service.
`socket_type`	Specify `stream` for a stream-based service, `dgram` for a datagram-based service, `raw` for a service that requires direct access to IP, and `seqpacket` for reliable sequential datagram transmission.
`protocol`	Specifies a protocol for the service. The protocol must exist in **/etc/protocols**. If this attribute is not defined, the default protocol employed by the service will be used.
`wait`	Specifies whether the service is single-threaded or multithreaded (`yes` or `no`). If `yes`, the service is single-threaded, which means that **xinetd** will start the server and then stop handling requests for the service until the server stops. If `no`, the service is multithreaded and **xinetd** will continue to handle new requests for it.
`user`	Specifies the user ID (UID) for the server process. The username must exist in **/etc/passwd**.
`group`	Specifies the group ID (GID) for the server process. The group name must exist in **/etc/group**.
`instances`	Specifies the number of server processes that can be simultaneously active for a service.
`nice`	Specifies the server priority.
`server`	Specifies the program to execute for this service.
`server_args`	Lists the arguments passed to the server. This does not include the

	server name.
only_from	Controls the remote hosts to which the particular service is available. Its value is a list of IP addresses. With no value, service is denied to all remote hosts.
no_access	Controls the remote hosts to which the particular service is unavailable.
access_times	Specifies the time intervals when the service is available. An interval has the form hour:min-hour:min.
log_type	Specifies where the output of the service log is sent, either the syslog facility (**SYSLOG**) or a file (**FILE**).
log_on_success	Specifies the information that is logged when a server starts and stops.
log_on_failure	Specifies the information that is logged when a server cannot be started.
rpc_version	Specifies the RPC version for a RPC service.
rpc_number	Specifies the number for an UNLISTED RPC service.
env	Defines environment variables for a service.
passenv	The list of environment variables from **xinetd**'s environment that will be passed to the server.
port	Specifies the service port.
redirect	Allows a TCP service to be redirected to another host.
bind	Allows a service to be bound to a specific interface on the machine.
interface	Synonym for **bind**.
banner	The name of a file to be displayed for a remote host when a connection to that service is established.
banner_fail	The name of a file to be displayed at the remote host when a connection to that service is denied.
groups	Allows access to groups the service has access to (**yes** or **no**).
enabled	Specifies the list of service names to enable.
include	Inserts the contents of a specified file as part of the configuration file.
includedir	Takes a directory name in the form of **includedir /etc/xinetd.d**. Every file inside that directory will be read sequentially as an **xinetd** configuration file, combining to form the **xinetd** configuration.

Table 3-8: Attributes for xinetd

You can start, stop, and restart **xinetd** using the service script in the **/etc/rc.d/init.d** directory and the **service** command, as shown here:

```
service xinetd stop
service xinetd start
service xinetd restart
```

On Fedora, you can also turn on and off particular **xinetd** services with **chkconfig**, as described earlier. Use the **on** and **off** options to enable or disable a service; **chkconfig** will edit the disable option for the service, changing its value to "yes" for off and "no" for on. For example, to enable the swat server, you could enter

```
chkconfig swat on
```

You can also use system-config-services to enable or disable a particular **xinetd** services. You cannot stop or start these services, just disable or enable them. To perform the equivalent of a start or start, you disable or enable, and then choose the xinetd service, and click the Restart button to restart the **xinetd** service. Restarting the **xinetd** service restarts the entire xinetd server, with your selected xinetd services disabled and enabled.

xinetd Configuration: xinetd.conf

The xinetd.conf file contains settings for your xinetd server, such as logging and security attributes. This file can also contain server configuration entries, but on Fedora these are placed into separate configuration files located in the **/etc/xinetd.d** directory. The **includedir** attribute specifies this directory:

```
includedir /etc/xinetd.d
```

Logging xinetd Services

You can further add a variety of other attributes such as logging information about connections and server priority (**nice**). In the following example, the **log_on_success** attribute logs the duration (**DURATION**) and the user ID (**USERID**) for connections to a service, **log_on_failure** logs the users that failed to connect, and **nice** sets the priority of the service to 10.

```
log_on_success += DURATION USERID
log_on_failure += USERID
nice = 10
```

The default attributes defined in the defaults block often set global attributes such as default logging activity and security restrictions: **log_type** specifies where logging information is to be sent, such as to a specific file (**FILE**) or to the system logger (**SYSLOG**), **log_on_success** specifies information to be logged when connections are made, and **log_on_failure** specifies information to be logged when they fail.

```
log_type = SYSLOG daemon info
log_on_failure = HOST
log_on_success = PID HOST EXIT
```

xinetd Network Security

For security restrictions, you can use **only_from** to restrict access by certain remote hosts. The **no_access** attribute denies access from the listed hosts, but no others. These controls take IP addresses as their values. You can list individual IP addresses, a range of IP addresses, or a

network, using the network address. The **instances** attribute limits the number of server processes that can be active at once for a particular service. The following examples restrict access to a local network 192.168.1.0 and the localhost, deny access from 192.168.1.15, and use the **instances** attribute to limit the number of server processes at one time to 60.

```
only_from = 192.168.1.0
only_from = localhost
no_access = 192.168.1.15
instances = 60
```

The **xinetd** program also provides several internal services, including **services**, **servers**, and **xadmin**: **services** provides a list of currently active services, and **servers** provides information about servers; **xadmin** provides **xinetd** administrative support.

xinetd Service Configuration Files: /etc/xinetd.d Directory

Instead of having one large **xinetd.conf** file for all services, the service configurations are split it into several configuration files, one for each service. The directory is specified in **xinetd.conf** file with an **includedir** option. The **xinetd.d** directory holds **xinetd** configuration files for services such as SWAT. This approach has the advantage of letting you add services by just creating a new configuration file for them. Modifying a service involves editing only its configuration file, not an entire **xinetd.conf** file.

Configuring Services: xinetd Attributes

Entries in an **xinetd** service file define the server to be activated when requested along with any options and security precautions. An entry consists of a block of attributes defined for different features, such as the name of the server program, the protocol used, and security restrictions. Each block for an Internet service such as a server is preceded by the keyword **service** and the name by which you want to identify the service. A pair of braces encloses the block of attributes. Each attribute entry begins with the attribute name, followed by an assignment operator, such as **=**, and then the value or values assigned. A special block specified by the keyword **default** contains default attributes for services. The syntax is shown here:

```
service <service_name>
{
<attribute> <assign_op> <value> <value> ...
 ...
}
```

Most attributes take a single value for which you use the standard assignment operator, **=**. Some attributes can take a list of values. You can assign values with the **=** operator, but you can also add or remove items from these lists with the **=+** and **=-** operators. Use **=+** to add values and **=-** to remove values. You often use the **=+** and **=-** operators to add values to attributes that may have an initial value assigned in the default block.

Attributes are listed in Table 3-8. Certain attributes are required for a service. These include **socket_type** and **wait**. For a standard Internet service, you also need to provide the **user** (user ID for the service), the **server** (name of the server program), and the **protocol** (protocol used by the server). With **server_args**, you can also list any arguments you want

passed to the server program (this does not include the server name). If **protocol** is not defined, the default protocol for the service is used.

Disabling and Enabling xinetd Services

You can turn services on or off manually by editing their **xinetd** configuration file. Services are turned on and off with the **disable** attribute in their configuration file. To enable a service, you set the disable attribute to **no**, as shown here:

```
disable = no
```

You then have to restart **xinetd** to start the service.

```
service xinetd restart
```

If you want to turn on a service that is off by default, you can set its **disable** attribute to **no** and restart **xinetd**. The entry for the TFTP FTP server, **tftpd**, is shown here. An initial comment tells you that it is off by default, but then the **disable** attribute turns it on:

```
service tftp
{
     socket_type     = dgram
     protocol        = udp
     wait            = yes
     user            = root
     server          = /usr/sbin/in.tftpd
     server_args     = -s /tftpboot
     disable         = yes
     per_source      = 11
     cps             = 100 2
     flags           = IPv4
}
```

Note: You can also use **xinetd** to implement SSH port forwarding, should your system be used to tunnel connections between hosts or services.

TCP Wrappers

TCP wrappers add another level of security to **xinetd**-managed servers. In effect, the server is wrapped with an intervening level of security, monitoring connections and controlling access. A server connection made through **xinetd** is monitored, verifying remote user identities and checking to make sure they are making valid requests. Connections are logged with the **syslogd** daemon and may be found in **syslogd** files such as **/var/log/secure**. With TCP wrappers, you can also restrict access to your system by remote hosts. Lists of hosts are kept in the **hosts.allow** and **hosts.deny** files. Entries in these files have the format **service:hostname:domain**. The domain is optional. For the service, you can specify a particular service, such as FTP, or you can enter **ALL** for all services. For the hostname, you can specify a particular host or use a wildcard to match several hosts. For example, **ALL** will match on all hosts. Table 3-9 lists the available wildcards. In the following example, the first entry allows access by all hosts to the web service, **http**. The second entry allows access to all services by the **pango1.train.com** host. The third and fourth entries allow FTP access to **rabbit.trek.com** and **sparrow.com**:

```
http:ALL
ALL:pango1.train.com
ftp:rabbit.trek.com
ftp:sparrow.com
```

The **hosts.allow** file holds hosts to which you allow access. If you want to allow access to all but a few specific hosts, you can specify **ALL** for a service in the **hosts.allow** file but list the hosts to which you are denying access in the **hosts.deny** file. Using IP addresses instead of hostnames is more secure because hostnames can be compromised through the DNS records by spoofing attacks where an attacker pretends to be another host.

Wildcard	Description
ALL	Matches all hosts or services.
LOCAL	Matches any host specified with just a hostname without a domain name. Used to match on hosts in the local domain.
UNKNOWN	Matches any user or host whose name or address is unknown.
KNOWN	Matches any user or host whose name or address is known.
PARANOID	Matches any host whose hostname does not match its IP address.
EXCEPT	An operator that lets you provide exceptions to matches. It takes the form of *list1* **EXCEPT** *list2* where those hosts matched in *list1* that are also matched in *list2* are excluded.

Table 3-9: TCP Wrapper Wildcards

When **xinetd** receives a request for an FTP service, a TCP wrapper monitors the connection and starts up the **in.ftpd** server program. By default, all requests are allowed. To allow all requests specifically for the FTP service, you enter the following in your **/etc/hosts.allow** file. The entry **ALL:ALL** opens your system to all hosts for all services:

```
ftp:ALL
```

Tip: Originally, TCP wrappers were managed by the **tcpd** daemon. However, **xinetd** has since integrated support for TCP wrappers into its own program. You can explicitly invoke the **tcpd** daemon to handle services if you wish. The **tcpd** Man pages (**man tcpd**) provide more detailed information about **tcpd**.

Network Time Protocol, NTP

For servers to run correctly, they need to always have the correct time. Internet time servers worldwide provide the time in the form of the Universal Time Coordinated (UTC). Local time is then calculated using the local systems local time zone. The time is obtained from Internet time servers from an Internet connection. You have the option of using a local hardware clock instead, though this may be much less accurate.

Normally, the time on a host machine is kept in a Time of Year chip (TOY) that maintains the time when the machine is off. Its time is used when the machine is rebooted. A host using the Network Time Protocol, then adjusts the time using the time obtained from an Internet time server. If there is a discrepancy of more than 1000 seconds (about 15 minutes), the system administrator is

required to manually set the time. Time servers in the public network are organized in stratum level, the highest being 1. Time servers from a lower stratum obtain the time from those in the next higher level.

For servers on your local network, you may want to set up your own time server, insuring that all your servers are using a synchronized time. If all your servers are running on a single host system that is directly connected to the Internet and accessing an Internet time server, you will not need to set up a separate time server. You can use the **ntpdate** command to update directly from an Internet time server.

```
ntpdate  0.fedora.pool.ntp.org
```

If the servers are on different host systems, then you may want a time server to insure their times are synchronized. Alternatively, you could just use the **ntpdate** command to update those hosts directly at given intervals. You could set up a cron job to perform the ntpupdate operation automatically.

There is one package on the Fedora repository for both the NTP server and its documentation, **ntp**.

The documentation will be located in the /usr/share/doc/ntp-doc directory in Web page format.

```
/usr/share/doc/ntp*/index.html
```

The ntp server

The NTP server name is **ntpd** and is managed by the **/etc/init.d/ntp** script. Use the start, stop, and restart options to mange the server

```
service ntp start
```

Your host systems can then be configured to use NTP and access your NTP time server.

To check the status of your time server, you can use the **ntpq** command. With the **-p** option is displays the current status.

```
ntpq -p
```

The ntp.conf configuration file

The NTP server configuration file is **/etc/ntp.conf**. This file lists the Internet time servers that your own time server used to deterring the time. The default **ntp.conf** file is shown here. Check the **ntp.conf** Man page for a complete listing of the NTP server configuration directives.

The server directive specifies the Internet time server Internet address that your NTP server uses to access the time. There are default entries for the Fedora time servers, but you can add more server entries for other time servers.

```
server 0.fedora.pool.ntp.org dynamic
server 1.fedora.pool.ntp.org dynamic
server 2.fedora.pool.ntp.org dynamic
```

NTP access controls

Access control to the NTP server is determined by the restrict directives. An NTP server is accessible from the Internet, anyone can access it. You can specify access options and the addresses of hosts allowed access. The **default** option lets you specify the set of default options. The **noquery**, **notrust**, **nopeer**, and **nomodify** option deny all access. The notrust option will not trust hosts unless specifically allowed access. The nomodify option prevents any modification of the time server. The **noquery** option will not even allow queries from other hosts, unless specifically allowed.

```
restrict -6 default kod notrap nomodify nopeer noquery
```

Then the local user, users on the same host that is running the NTP server, is allowed to access the NTP server. Addresses are specified for both IPv4 and IPv6 local host, **127.0.0.1** and **::1**.

```
restrict 127.0.0.1
restrict ::1
```

To allow access from hosts on a private local network, you can use the restrict directive to specify the local network address and mask. The following allows access to a local network, 192.168.123, with a network mask of 255.255.255.0 to determine the range of allowable host addresses.

```
restrict 192.168.123.0 mask 255.255.255.0
```

If you want to require the use of encrypted keys for access, add the **notrust** option. Use **ntp-keygen** to generate the required encryption keys (symmetric or public). Check the NTP documentation and **ntp-keygen** man page for a detailed description encryption support and ntp encryption commands. Encryption can be symmetric or public (public/private keys). Symmetric keys are kept in the /etc/ntp/keys file with a key identifier to be used to reference the key.

```
keys /etc/ntp/keys
```

Public/private keys are enabled with the **crypto** directive and place their keys in the **/etc/ntp/crypto** directory. The password for decrypting keys is kept in the **/etc/ntp/crypto/pw** file.

```
includefile /etc/ntp/crypto/pw
```

You can also run the time server in broadcast mode where the time is broadcasted to your network clients (this can involve security risks). Use the broadcast directive and your network's broadcast address. Your host systems need to have the **broadcastclient** setting set, which will listen for time broadcasts.

```
broadcast 192.168.123.255
```

ntp.conf

```
## For more information about this file, see the man pages
# ntp.conf(5), ntp_acc(5), ntp_auth(5), ntp_clock(5), ntp_misc(5), ntp_mon(5).

driftfile /var/lib/ntp/drift

# Permit time synchronization with our time source, but do not
# permit the source to query or modify the service on this system.
restrict default kod nomodify notrap nopeer noquery
```

```
restrict -6 default kod nomodify notrap nopeer noquery

# Permit all access over the loopback interface.  This could
# be tightened as well, but to do so would effect some of
# the administrative functions.
restrict 127.0.0.1
restrict -6 ::1

# Hosts on local network are less restricted.
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool ().
server 0.fedora.pool.ntp.org dynamic
server 1.fedora.pool.ntp.org dynamic
server 2.fedora.pool.ntp.org dynamic

#broadcast 192.168.1.255 autokey     # broadcast server
#broadcastclient                     # broadcast client
#broadcast 224.0.1.1 autokey         # multicast server
#multicastclient 224.0.1.1           # multicast client
#manycastserver 239.255.254.254              # manycast server
#manycastclient 239.255.254.254 autokey # manycast client

# Undisciplined Local Clock. This is a fake driver intended for backup
# and when no outside source of synchronized time is available.
#server 127.127.1.0    # local clock
#fudge  127.127.1.0 stratum 10

# Enable public key cryptography.
#crypto

includefile /etc/ntp/crypto/pw

# Key file containing the keys and key identifiers used when operating
# with symmetric key cryptography.
keys /etc/ntp/keys

# Specify the key identifiers which are trusted.
#trustedkey 4 8 42

# Specify the key identifier to use with the ntpdc utility.
#requestkey 8

# Specify the key identifier to use with the ntpq utility.
#controlkey 8

# Enable writing of statistics records.
#statistics clockstats cryptostats loopstats peerstats
```

NTP clock support

You can also list a reference to the local hardware clock, and have that clock used should your connection to the Internet time server fail. The hardware clock is references by the IP address that has the prefix 127.127 followed by the clock type and instance, as in 127.127.1.1. The type for the local clock is 1.

```
server 127.127.1.0
```

The **fudge** directive is used to specify the time for a hardware clock, passing time parameters for that clock's driver.

fedora

4. Installing and Updating Software

Fedora software has grown so large and undergoes such frequent updates that it no longer makes sense to use discs as the primary means of distribution. Instead distribution is effected using an online Fedora software repository. This repository contains an extensive collection of Fedora-compliant software.

For Fedora, you can add software to your system by accessing software repositories supporting YUM (Yellowdog Update, Modified). In addition many software applications, particularly multimedia ones, have potential licensing conflicts. By leaving such software in third party repositories, Fedora avoids possible legal issues. Many of the popular multimedia application such as video and digital music support can be obtained from the third party repository RPM Fusion, using the same simple YUM commands you would use for Fedora sponsored software.

This approach heralds a move from thinking of most Linux software included on a few disks, to viewing the disk as just a core from which you can expand your installed software as you like from online repositories. Most software is now located on the Internet connected repositories. With the integration of YUM into your Fedora system, you can now think of that software as an easily installed extension of your current collection. You can find out more about how YUM is used on Fedora in Managing Software with YUM, located on the Fedora documentation page:

http://docs.fedoraproject.org/yum/en/

Updating Fedora: Update System (PackageKit)

New versions of Fedora are released every six months or so. In the meantime, new updates are continually being prepared for particular software packages. These are posted as updates you can download from software repositories and install on your system. These include new versions of applications, servers, and even the kernel. Such updates may range from single software packages to whole components—for instance, all the core, application, and development packages issued when a new release of GNOME, KDE, or X11 is made available.

TIP: If you are installing or updating several months after the official Fedora release, it may be easier to use a Fedora Re-Spin disk. This is a disk with the latest updates. This way you avoid having to download numerous updates to the original packages from online repositories. Fedora Re-Spins are issued by the Fedora Unity project. Fedora Re-Spins disks can be downloaded from **www.fedoraunity.org**.

Updating your Linux system has become a very simple procedure, using the automatic update tools. For Fedora, you can update your system by accessing software repositories supporting the YUM (Yellowdog Updater, Modified) update methods. YUM uses RPM headers to determine which packages need to be updated. To update your packages, you now use the update capability of PackageKit, labeled as Update System in the System | Administration menu. PackageKit is the graphical interface for YUM, which now performs all updates. With Update System (PackageKit) you no longer have to update using a **yum** update command entered in a terminal window.

PackageKit makes use of the Update System applet on your GNOME panel, which will automatically check for updates whenever you log in. If updates are detected, PackageKit will display an update icon on the panel and display a message in the lower right panel telling you that updates are available and how many there are (see Figure 4-1). The message has buttons to install updates and a button to not show the message again. If you do not choose to install immediately, you can close the window and then later choose to install using the update menu accessible from

the update system icon. Click the update system icon to opens a menu (see Figure 4-1) where you can select either to perform all updates directly (Update system Now), or to review and possibly choose updates to perform (Show Updates). If you choose to Update System Now, a small progress bar is displayed under the applet as the downloads and updates are performed.

The update icon image shown varies on whether the update includes security, bug, or software enhancement updates. All three could be included in an update, but for the icon display, security takes precedent, followed by bugs, and then enhancements. If the update includes any security fixes, the red star security icon is shown. If there are no security updates, but there are bug fixes, then the orange star with a bug image is used. For just enhancements, then the orange star is displayed.

Figure 4-1: Update System (PackageKit) notification message and update icon and menu

The Show Updates option will start up the **gpk-update-viewer** tool to display your updates. You can also select Update System manually from its Update System entry in the System | Administration menu. Initially the Update Overview window is displayed (see Figure 4-2).

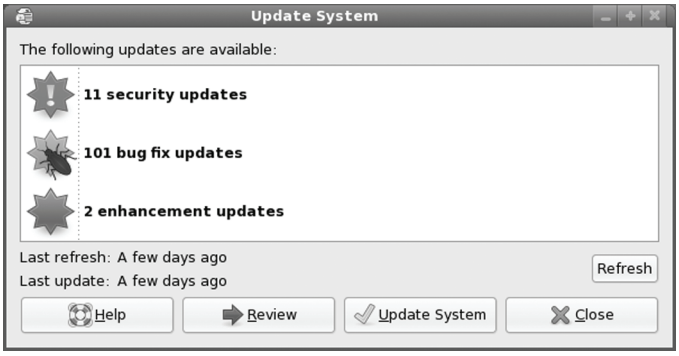


Figure 4-2: Update System with available updates, Overview window

The number of updates is shown for each category: security, bugs, and enhancements. The last refresh and the time of the previous update are also displayed. There are buttons to update the system or review the packages to update, "Update System" and "Review". To simply perform the

updates, click the Update System button. If you want review the updates, and, perhaps deselect certain updates, while deselecting others, you can click the Review button.

Tip: After a new install, if Update System fails to run or function, you may need to first update using the **yum** command. Enter the **yum update** command in a terminal window (Applications | System Tools). Enter y at the prompts. This initial update should fix problems with Update System, which should then function.

On the Review window, all needed updates will be selected automatically when Update System starts up. The check boxes for each entry let you deselect any particular packages you may not want to update (see Figure 4-3). Should you want to see details about a particular package, click on the package entry in the package listing. The window is then divided into two panes, a package listing and package description. Information about the package will display on the lower pane (see Figure 4-4). Information includes features like the version and repository. Bug fixes will show Bugzilla entries for bugs that have been fixed. Software updates will show information about the update, including a description of new features.

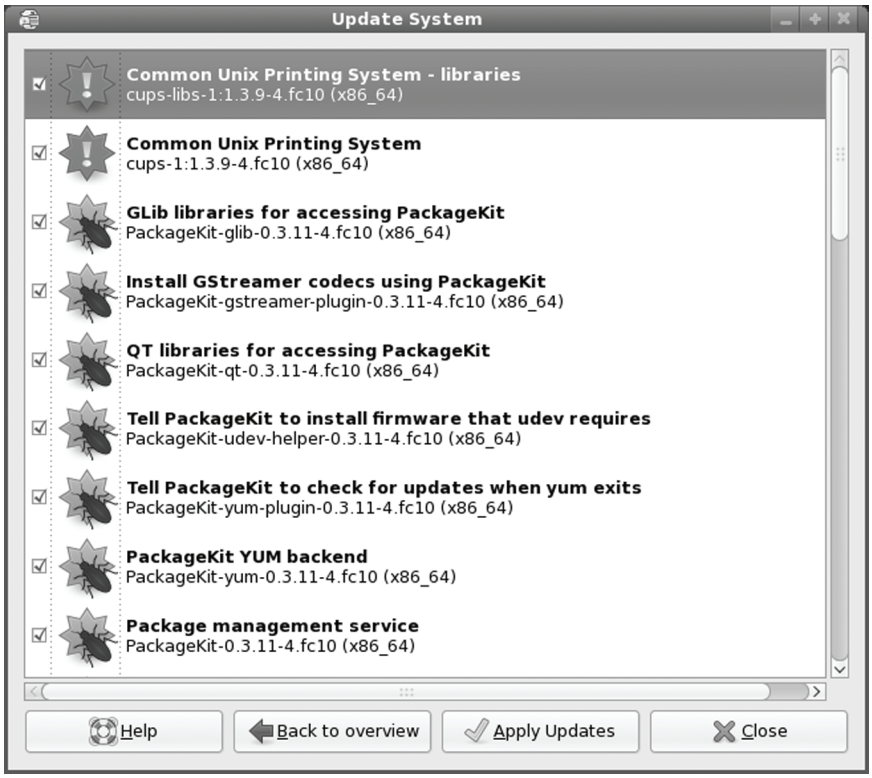


Figure 4-3: Package Updater listing packages to be updated, Review window

Click the Apply Updates button to start updating. An Authenticate dialog will first be displayed prompting you to enter your root password, allowing the updates to be performed (see Figure 4-5).

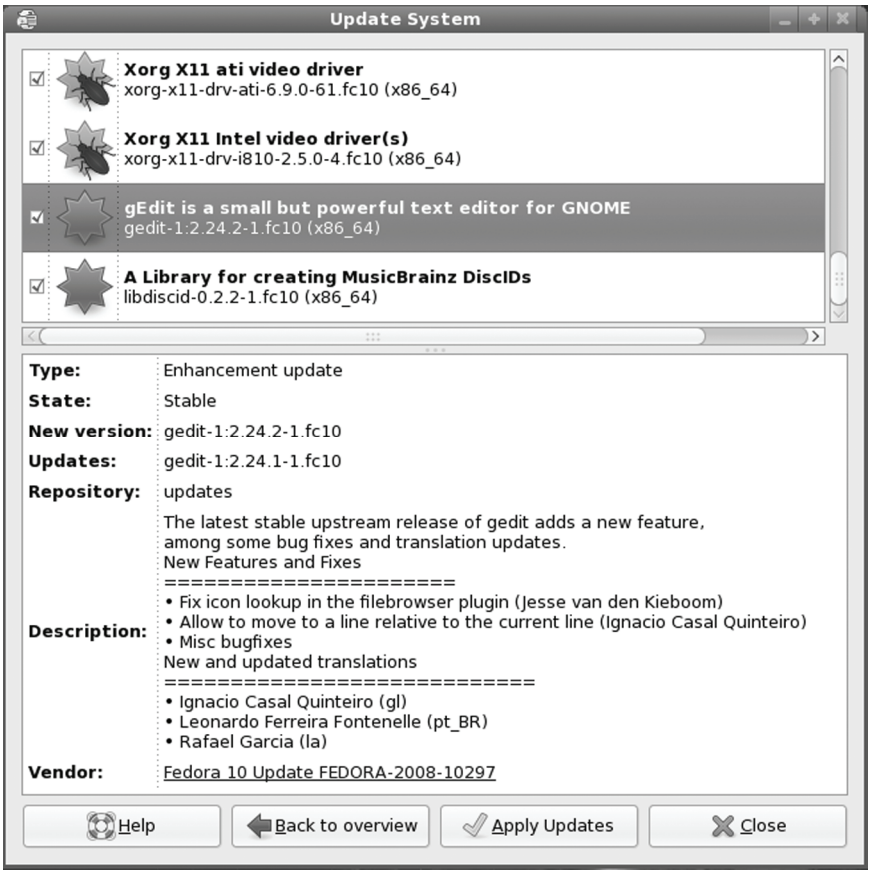


Figure 4-4: Package Updater selected package with displayed description

Figure 4-5: Package Updater authorization

During the update, a status window will open displaying the different update stages, beginning with dependency checks, then download, followed by testing, and ending with update, installation, and cleanup (see Figure 4-6). You can click the Cancel button during dependency and downloading to cancel the update. The Close button will close the window, but continue with the updating. The Help button opens the PackageKit manual.

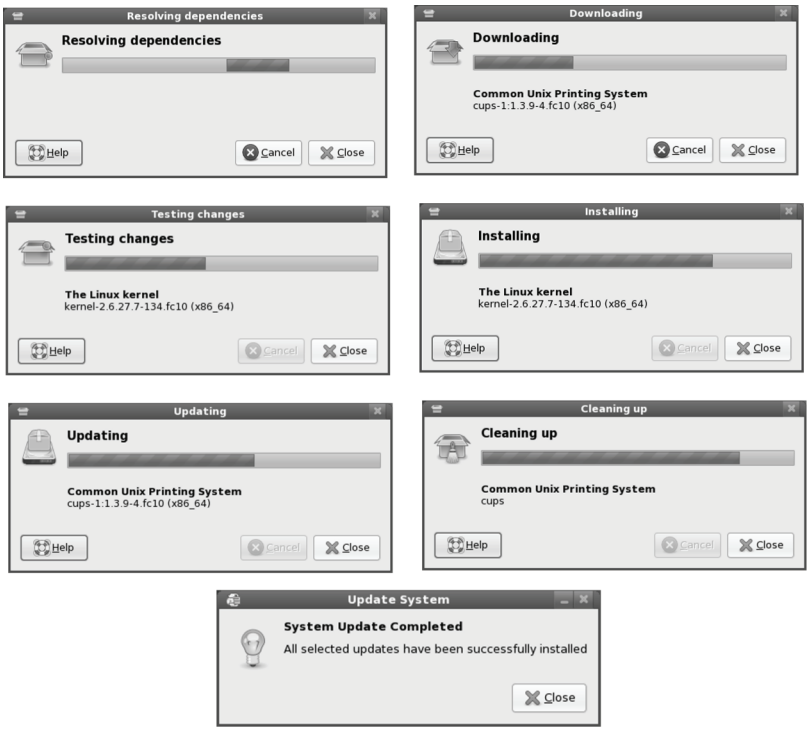


Figure 4-6: Update progress window

If, during update, Packagekit has to access a new repository for the first time, you will be prompted to accept the repository's key (see Figure 4-7). Click the Yes button to approve access to that repository. You will then be prompted to authenticate the action by entering the root user (administrative) password.

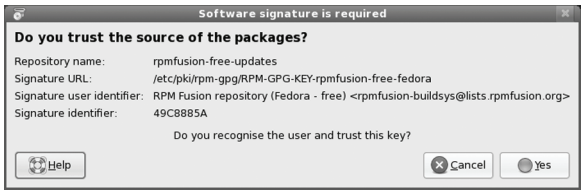


Figure 4-7: New repository key acceptance prompt

Should a new kernel be installed, when you boot you will then be using the new kernel. Your old kernel will remain as a GRUB option should you have difficulties with the new kernel. To choose the old kernel, you will have to select it from the GRUB boot screen (press the spacebar on start up). To make the old kernel the default again, you can either select it with the Bootloader configuration screen (System | Administration | Bootloader) or set the default option in the **/etc/grub/grub.conf** file to 1 (the new kernel will be 0).

Tip: To upgrade to a new Fedora release, you can download Fedora DVD install ISO image through http://fedoraproject.org/get-fedora. You can then burn the ISO

image and upgrade from that install DVD. Alternatively, you can use the **yum** command with the **upgrade** option in any terminal window. This will perform a package by package online upgrade.

Manual Update

You can also manually perform updates instead of waiting for a notification icon. Select Update System from the System | Administration menu. This opens the Update System Overview window with a notice showing how many updates are available, along with the time since the last refresh and the last update (see Figure 4-2)

To perform your package updates, you can just click the Update System button.

A refresh downloads the update package listings from repositories showing packages to update. A refresh will occur automatically at scheduled times. You can manually perform a refresh by press the Refresh button. If it has been more than several hours or days since your last refresh, you should perform one before updating.

To see a listing of updates, click the review button. This displays the Update System review window, as shown in Figure 4-3), with a listing of packages to update. You can deselect those you do not want updated. You can then click the Update System button to perform the updates. The overview button returns you to the Overview window (Figure 4-2), showing the number of updates and the times of the last refresh and update.

Update Preferences

You can choose when checks are made for updating as well as whether to perform the updates automatically. Use the Update Preferences window to set your update options. Select System | Preferences | Software Updates. This opens the Software Update Preferences window where you can set the intervals for checking for updates and major upgrades as well as whether to enable automatic updating (see figure 4-8). You can also specify when the Update icon will be displayed on the top panel.

Initially, updates will be checked daily and no action will be taken. The user will have to apply the updates. On the Check for update pop up menu you can select hourly, daily, weekly, or never. Daily is selected and is the option normally used. The never option would require that the user manually check for updates.

The Automatically install pop up menu will have options for All updates, Only security updates, or Nothing. If you want all software and security updates to be performed automatically for you, select All updates. Should you want to be selective as to what applications you update, you could select Only security updates.

On the "Check for major upgrades" pop up menu you select Daily, Weekly, or Never.

The Display Notification section has options for notifying you when updates are available, and when long tasks are completes. If you selected All updates for the Automatically install setting, then the When updates are available option in the Display Notification section will be grayed out.

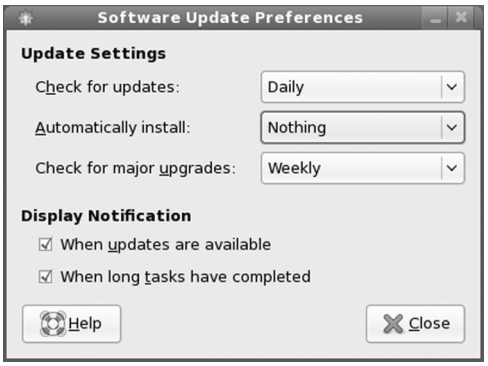


Figure 4-8: Software Update Preferences

Update with the yum command

Alternatively you can update using the **yum** command with the **update** option. The following would update an already installed **Gnumeric** package.

```
yum update gnumeric
```

You can use the **check-update** option to see what packages need to be updated.

```
yum check-update
```

To perform a complete update of all your installed packages, you just use the update option. This would have the same effect of updating with Update System.

```
yum update
```

In some cases, when you first install a new system from your CD/DVD, there may be incompatibilities or bugs that prevent you from using Update System to update to the latest system fixes on the Fedora repository. In this case you can open a terminal window and update directly with the **yum** command.

You may also need to perform a **yum** command update should you have display problems and you cannot access your desktop. An update could fix the display problem. You could then boot into the command line interface (runlevel 3), and, after login in, enter the **yum update** command.

A manual update may also become necessary if you have broken dependencies. If, among a set of packages to be updated, just one or more have broken dependencies, then the entire set will not be updated by PackageKit. PackageKit does not always inform you of the packages causing trouble. Otherwise you could just deselect them from the package listing in Update System. To exclude a package you can open a terminal window and run **yum update** with the **--exclude** option. To select several packages with the same prefix, add a ***** file matching operator to the package name. You can add as many exclude options as you want. The following updates all packages except the Perl packages.

```
yum update --exclude=perl*
```

Automatic YUM Update with cron

You can also setup up YUM to perform automatic updates using cron. The **yum-cron** package installs cron configuration file for YUM. These include a **yum.cron** files in the **/etc/cron.daily** and **/etc/cron.weekly** directories, which will automatically update your system. The cron entry will first update the YUM software if needed and then proceed to download and install any updates for your installed packages. It runs **yum** with the **update** option. You can set up update options like download only in the **/etc/sysconfig/yum-cron** file.

The automatic update will run only if it detects a YUM lock file in the **/var/lock/subsys** directory. By default, this is missing. You can add it using the **yum-cron** service script. The **start** option creates the lock file, enabling the cron-supported updates, and the **stop** option removes the file, disabling the automatic update.

```
yum-cron start
```

Installing Software Packages

Installing software is an administrative function performed by a user with administrative access. Unless you chose to install all your packages during your installation, only some of the many applications and utilities available for users on Linux were installed on your system. On Fedora, you can easily install or remove software from your system with either the PackageKit software manager (Add/Remove Software), the **yum** command, or the **rpm** command. Alternatively, you can install software by downloading and compiling its source code. The procedure for installing software using its source code has been simplified to just a few commands, though you have a great deal of flexibility in tailoring an application to your specific system.

An RPM software package operates like its own installation program for a software application. A Linux software application often consists of several files that must be installed in different directories. The program itself is most likely placed in a directory called **/usr/bin**, online manual files go in another directory, and library files go in yet another directory. In addition, the installation may require modification of certain configuration files on your system. The RPM software packages perform all these tasks for you. Also, if you later decide you don't want a specific application, you can uninstall packages to remove all the files and configuration information from your system.

The software packages on your DVD/CD-ROMs, as extensive as they are, represent only some of the software packages available for Fedora Linux. Most reside on the Fedora Software repository, a repository whose packages are available for automatic download using YUM and its front ends, PackageKit or the **yum** command. Many multimedia applications and support libraries can be found at the RPM Fusion repository (**http://rpmfusion.org**), and, once configured, downloaded directly with YUM and PackageKit. Table 4-1 lists several Fedora software sites. Fedora and RPM Fusion are all YUM supported, meaning that a simple YUM configuration enables you to directly download and install software from those sites using the **yum** command or the PackageKit software manager. Fedora software repository YUM configuration is already installed, and RPM Fusion provides its own YUM configuration file. Check **http://docs.fedoraproject.org/yum/en/** for information on using YUM.

Fedora provides only open source applications in its own repository. For proprietary applications like NVIDIA's own graphics drivers or multimedia application that may have patent

issues, you need to use the third-party repository RPM Fusion. The list of forbidden items for the official Fedora repository can be found at **www.fedoraproject.org/wiki/ForbiddenItems**. These include items like the NVIDIA and ATI graphics drivers (those you can obtain from **http://rpmfusion.org**).

Most software available for Fedora will be on either the Fedora or RPM Fusion repositories. There may be a few that you want that available on other repositories. Usually these other repositories provide only RPM package files, not YUM configuration. Software sites for additional software include GNOME's **www.gnomefiles.org**, KDE's **www.kde-apps.org**, and **www.sourceforge.net**. The **www.sourceforge.net** site not only distributes software but also serves as the primary development site for a massive number of open source software projects. You can also locate many of the newest Linux applications from **www.freshmeat.net** or **www.rpmfind.net**. Check first, though, to see if they are already available on the Fedora or RPM Fusion repositories, most are.

Internet Sites	Description
`http://fedoraproject.org/wiki/Distribution/Download`	Fedora download instructions and distribution information
`http://fedoraproject.org/get-fedora`	Fedora distribution disks, download links for all formats
`http://torrents.fedoraproject.org`	Fedora BitTorrent files for ISO and DVD disks
`http://download.fedoraproject.org`	Fedora Software repository, mirror link
`http://download1.rpmfusion.org` `http://rpmfusion.org`	RPM Fusion repository for third-party multimedia and driver Fedora compliant software (Fedora Project extension)
`http://fedoraproject.org/wiki/ForbiddenItems`	Packages not included in the main Fedora software repository.
`http://fedoraproject.org/wiki/Multimedia`	Information on multimedia packages available for Fedora

Table 4-1: Fedora Software Sites

Installing with YUM

Downloading Fedora software or software from any configured Fedora YUM repository is a simple matter of using Add/Remove Software (PackageKit), which provides a GUI interface for YUM. Alternatively, you can enter the **`yum`** command with the **`install`** option and the name of the package in a terminal window. YUM will detect the software and any dependencies, and it will prompt you to download and install it. For example, the following command will install Abiword:

```
yum install abiword
```

Installing individual packages with your browser

Alternatively, you can download an individual package directly using your browser. You would use this method only for packages not already available from YUM supported repositories. You will also have to manually install any needed dependent packages, as well as check system compatibility (x86_64, i38c, ppc). Your Fedora Web browsers will let you perform both a download and install in one simple operation, invoking PackageKit to perform the install (**gpk-install-file**). On a GNOME desktop already-downloaded packages can be installed with a simple right-click and install selection, invoking PackageKit to perform the installation.

PackageKit, Add/Remove Software

PackageKit is the software management front end for YUM. The PackageKit Package Manager is Internet based, installing from online repositories, using YUM to download and install. It is designed to be a cross-distribution package manager that can be used on any YUM supported Linux distribution.

PackageKit provides a variety of applications for different software tasks, including installation, removal; updating, repository management, logs, and software install of a package file (see Table 4-2). The primary PackageKit application is **gpk-application**, access as Add/Remove Software on the System | Administration menu. The **gpk-update-viewer** lets you examine and select updates. The **gpk-log** (Applications | System Tools | Software Log) display a list of all your previous install, removal, and update operation, including major updates and individual package installs.

To use PackageKit you select the Add/Remove Software entry from the System | Administration menu. PackageKit will start up by gathering information on all your packages. An Add/Remove Software window opens with a sidebar for searching and a category list. Before you install packages, it is advisable to first refresh your software lists. This is the listings of software packages available on your enabled repositories. Select Refresh applications lists from the System menu.

Application	Description
`gpk-application`	Add/Remove Software
`gpk-update-viewer`	Update you system, Update System
`gpk-prefs`	Configure update preferences, Preferences | System | Software Updates.
`gpk-repo`	Manage software repositories
`gpk-log`	history of updates, Administration | System Tools
`gpk-install-file`	Install local software packages
`gpkupdate-icon`	display update icon

Table 4-2: PackageKit applications

PackageKit Browsing

You can find packages by category by selecting a category on the sidebar. All the packages in that category will be listed. Uninstalled packages will be light-shaded with a closed box icon. Figure 4-9 shows the packages for the Gnome desktop category. Using categories is similar to the Browse panel in the Pirut Add/Remove Software utility used in previous Fedora releases. Categories let you browse through your software, seeing what is available for different kinds of tasks or features, like Multimedia applications or applications designed for the GNOME desktop.

Figure 4-9: Add/Remove Software, PackageKit

To install several packages at once, click on their checkboxes, and then click the Apply button. The status bar at the bottom will show tasks being performed, starting with downloading, then installing, and finally finished. A progress bar will show the install tasks progress. The PackageKit notification icon in the top panel will also show a download image as the package is downloaded. Once installed, the package icon color will change from a light shade to a solid one and be replaced by the open box icon.

PackageKit Selected Package Actions and Information

To see information about a particular package, select it. This opens a pane at the bottom with panes for Description and the Project home page (see Figure 4-10). A Selection menu will appear on the menubar between the Filters and Help menu buttons (see Figure 4-11). Click on the Selection button to display a menu with several action and information choices. The "Project homepage" entry will open your browser to the package home page where you can find more detailed information about it. The "Run program" entry lets you run an installed package directly.

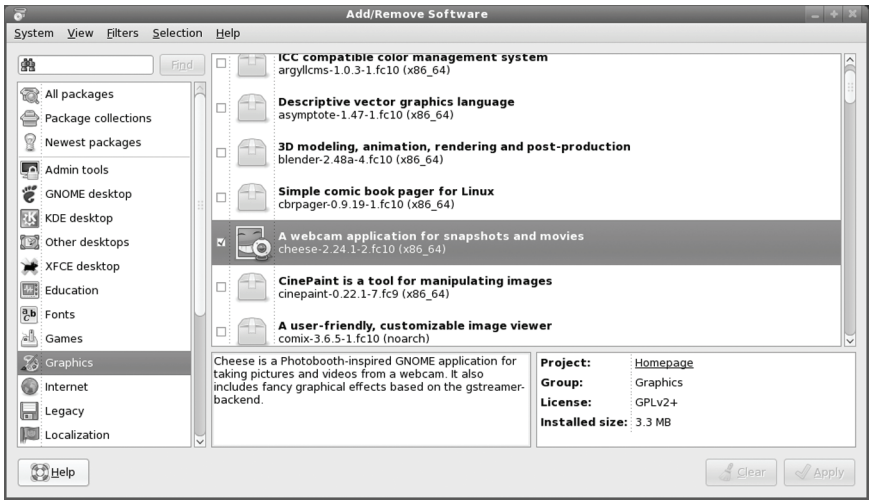


Figure 4-10: Package information

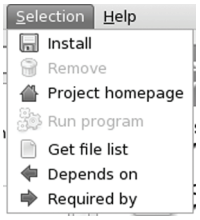


Figure 4-11: Package Selection Menu

You can also install or remove a selected package. Just click on the package and then, from the Selection menu, choose Install or Remove. For an uninstalled package, choosing Install from the Selection menu will place a checkmark for you on the selected package's checkbox. To actually perform the install, you then click the Apply button. An installed package will only allow removal, and an uninstalled package allows only installation. For installed packages, choosing Remove will uncheck the package's checkbox. Clicking Apply will then remove the package. Before you click Apply, you can always change your mind, checking or unchecking the package's checkbox as you wish.

The "Get file list", "Depends on", and "Required by" entries will open a separate window listing those files and packages. The "Get file list" entry opens a window that lists all the actual files the package will install, or has installed (see Figure 4-12). This can be helpful for tracking down the actual command names and location of configurations files.

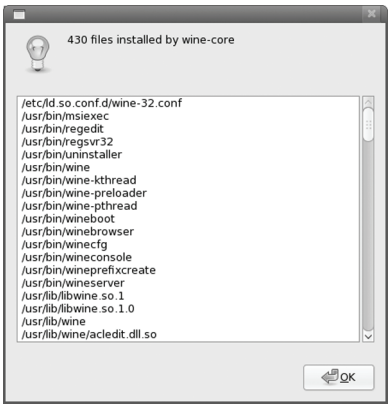


Figure 4-12: Package Selection, "Get file list"

The "Depends on" and "Required by" entries open windows that show the package's dependencies, those that depend in it as well as those it, in turn, depends on (see Figure 4-13).

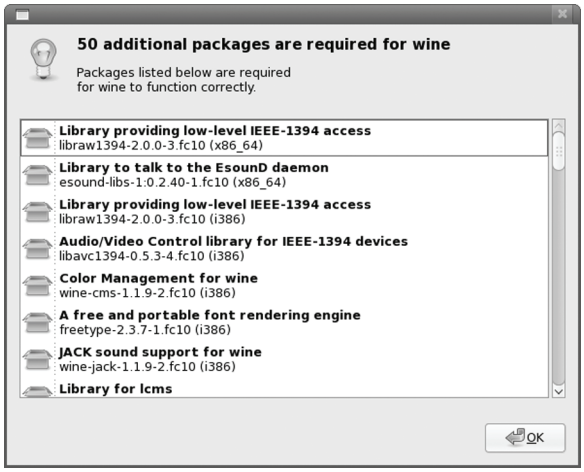


Figure 4-13: Package Selection, "Depends on"

PackageKit Searching

Instead of tracking down a package through categories, you can just search for it using its package name. Just enter a pattern to search for in the Search box, located at the top of the category sidebar, and press ENTER or click the Find button. The results will be listed as shown in Figure 4-14. Here the pattern searched for is **win**, listing all the Fedora packages beginning with *win*, including the wine packages. Select a package to then install it, showing its information panels and Install button. The status bar will show download and install tasks with a progress bar as they are performed.

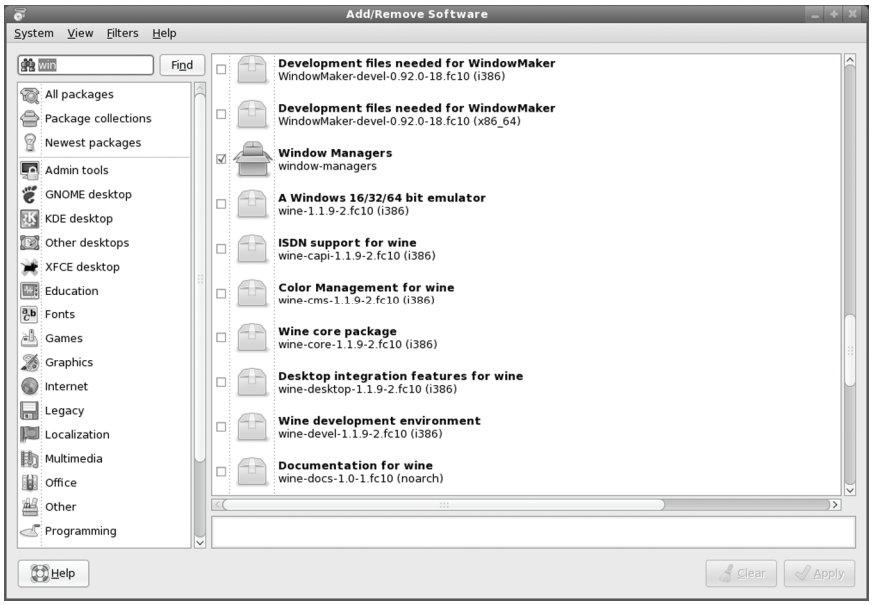


Figure 4-14: Package searching

If you enter a complete package name, like wine, that package will be selected and displayed (see Figure 4-15).

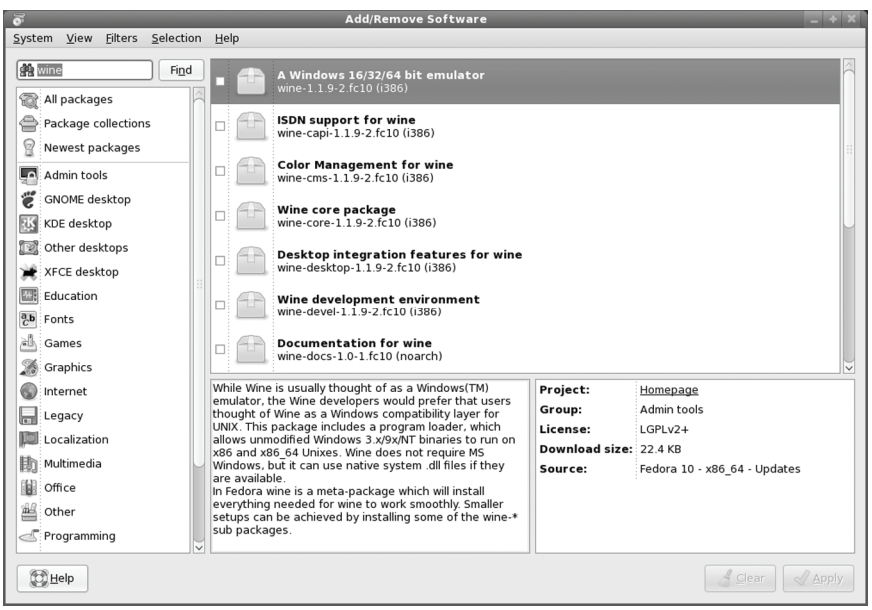


Figure 4-15: Package searching with exact match

PackageKit Filtering

Instead of showing all your available packages, both installed and uninstalled, you can filter the package listing. Several filters are available from the Filters menu (see Figure 4-16). You can filter by Installed, Development, Graphical, and Free packages. The Installed option lets you display only installed or uninstalled packages (available). Graphical is for packages that operate on a GUI desktop. The Free option will let you choose only open source free packages, instead of any licensed or restricted proprietary packages. Restricted packages like graphic drivers, could still be freely available, but not open source, and in that sense not free. If you are looking for software you know is not free, you can select the "Only non free" option.

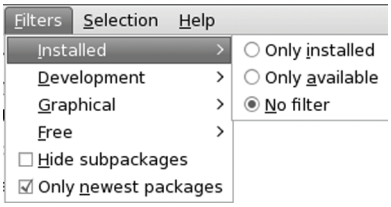


Figure 4-16: Package Filters

Managing Repositories

YUM is the primary install packages tool. When you install a package, YUM will be invoked and it will automatically select and download the package from the appropriate repository. After having installed your system, when you then want to install additional packages, PackageKit will use YUM to install from a repository. This will include all active YUM online repositories you may have configured such as sites like RPM Fusion, not just the Fedora and update repositories configured for you during installation.

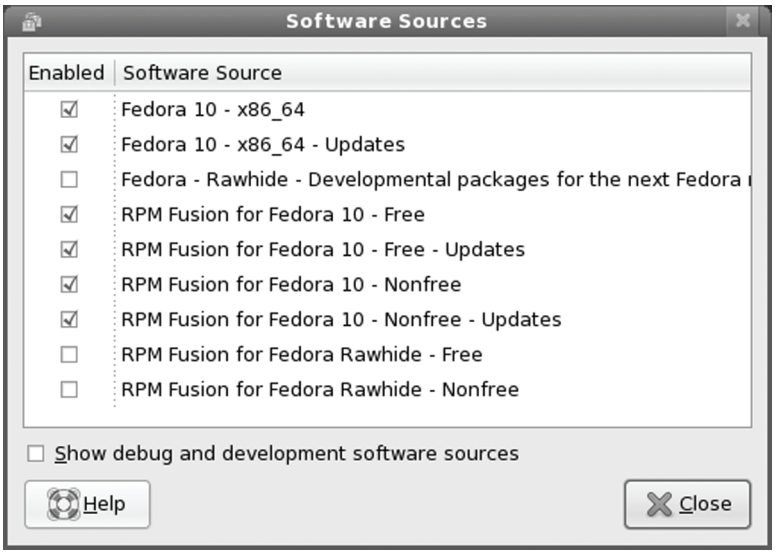


Figure 4-17: Software Sources: Configured Software Repositories

You can enable or disable your configured YUM repositories using the Repository Viewer (**gpk-viewer**). On the Add/Remove Software window, select Software Sources from the System menu. You can also select Software Sources from the System | Administration menu. This opens the Repository Viewer window listing all your configured repositories (see Figure 4-17).

The repository viewer will lists all configured repositories. These are repositories that have YUM configuration files in the **/etc/yum.repos.d** directory, holding information like URLs, GPG keys, and descriptions. Those that are enabled will be checked. You can disable a repository by un-checking it. The Fedora repository names will include the release number and the platform, like x86_64 for the 64 bit version. Fedora and Fedora Update repositories will usually be enabled.

Add/Remove Software will list the packages available from all the active repositories. If you should want to see just the package from one or the other repository, you could deactivate the others. For example, if just wanted to see what was on the RPM Fusion repository, you could deactivate all others, including Fedora. Then the only packages on the RPM Fusion repository would be shown. You can later reactivate the other repositories.

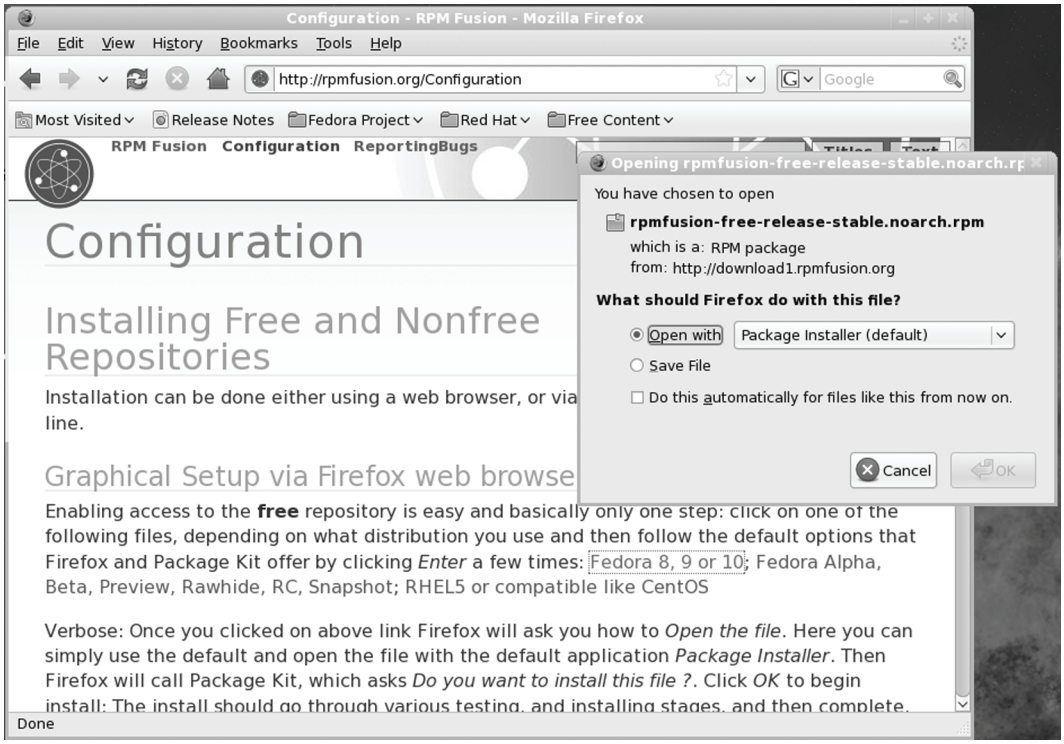


Figure 4-18: RPM Fusion repository configuration file downloaded from http://rpmfusion.org

Repositories also have specialized repositories for development, debugging, and source code files. Here you have find applications under development that may be useful, and well as the latest revisions. Some will have testing repositories for applications not yet completely validated for the current release. These applications may or may not work well. Source, Test, and Debug

repositories will normally be disabled. Rawhide is the development repository for a future release currently under development. The source repositories hold source code packages, and the test repositories are for packages still in the testing stages.

Note: To remove packages with PackageKit, just locate and select its entry, and then click the Uninstall button.

Using the RPM Fusion repository with PackageKit

The RPM Fusion repository (**http://rpmfusion.org**) holds most of the third party drivers and codecs most users want for multimedia tasks. These include the vendor graphics drivers for Linux, such as those provided by Nvidia and ATI/AMD. The RPM Fusion packages are configured for Fedora. Also included on the repository are multimedia codecs with licensing issues like MP3, AC3, and MPEG2 codecs, as well ad MPEG4 and DivX (Xvid).

To access the RPM Fusion repository, you first have to download and install both its YUM configuration files and its GPG authentication key. These are both included in RPM Fusion's **rpmfusion-free** package. Simply download and install this package using your Web browser. The package is located at **http://rpmfusion.org** (see Figure 4-18). Click on the Configuration link at the top of the page to go to the Configuration page. On the section titled "Graphical Setup via Firefox web browser", click on the links labeled "Fedora 8, 9, or 10. There are free and nonfree packages, with links for each, **rpmfusion-free** and **rpmfusion-nonfree**. When you click the link to download, Fedora detects that it is a software package and opens a dialog with the option to install it directly (see Figure 4-18). Be sure to install the **rpmfusion-free** package first. The nonfree package is optional, but requires that the free package already be installed.

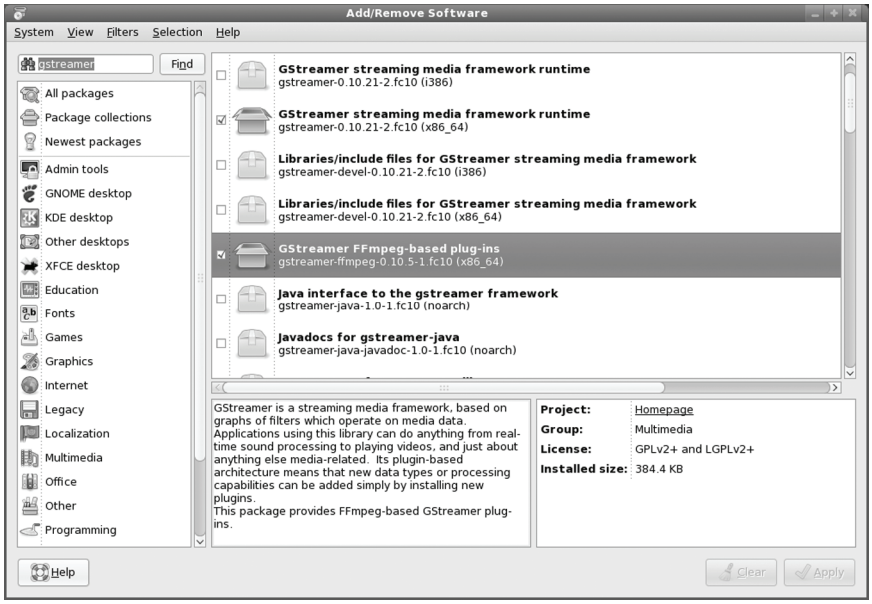


Figure 4-19: Software package accessible from RPM Fusion repository

Perform the installation. You will be warned that there is no authentication key for the package. Install anyway. The key is not yet installed, but will be as part of the package.

The RPM Fusion Configuration Web page also tells you how to perform an installation of both free and nonfree RPM Fusion YUM configuration packages from the command line using the **rpm** command.

Once installed, you can open Software Sources to check that the RPM Fusion repository is enabled (see Figure 4-19), System | Administration | Software Sources.

With RPM Fusion repository now enabled, you can use PackageKit to search, select, and install any packages on its repository. Both RPM Fusion and the Fedora repositories will have their packages intertwined in the PackageKit display (see Figure 4-19).

Note: It is still possible to use the atrpms repository (**http://atrpms.net**), though most of its packages, including MythTV, are now available from the RPM Fusion repository. There are potential conflicts between atrpms packages and RPM Fusion packages. Most packages are duplicated on the RPM Fusion repository. As a rule, you should only use RPM Fusion packages not on the atrpms packages.

YUM Extender: yumex

The YUM Extender (Yumex) is an alternative GUI interface for managing software packages on your YUM repositories (see Figure 4-20). Yumex is included with Fedora. The YUM Extender provides more flexibility than PackageKit. It provides detailed package information, as well as allowing you to turn repositories on an off directly, without having to edit repo files. Once installed, you can access Yumex with the YUM Extender entry in the Applications | System Tools menu. The YUM Extender screen has five panels, each accessed by view icons on the left column, as shown in Figure 4-20. You can select package using either the Package or the Groups button. The Package button will list all possible packages, whereas the Groups button shows the categories. Uninstalled packages are shown in red.

The Package pane features ways to narrow your displayed packages. You can choose to display updates, installed packages, or uninstalled packages only. You can further specify categories, like repositories, displaying only packages from a certain repository. Clicking a package will display detailed information about it on the panel below, where you can select the description, files included with the package, and the change log if available.

To select a package, just check the one you want to install. Once selected, the package is added to the Process Queue panel. Once you have added packages to the Queue, click the Queue icon to see the list of packages you want to install. From this panel you can choose to delete items or just save the list for later installation. To perform the installation, click the Process Queue button. All dependent packages will also be displayed. The install process will be shown on the Output panel.

The Repos panel operates as a repository manager, listing all your configured repositories. From this panel you can enable or disable any repositories. For example, if you need to temporarily enable the RPM Fusion repository, you can select it on this panel, and then install software from that repository. Once finished, you could then disable the repository using the Repos panel.

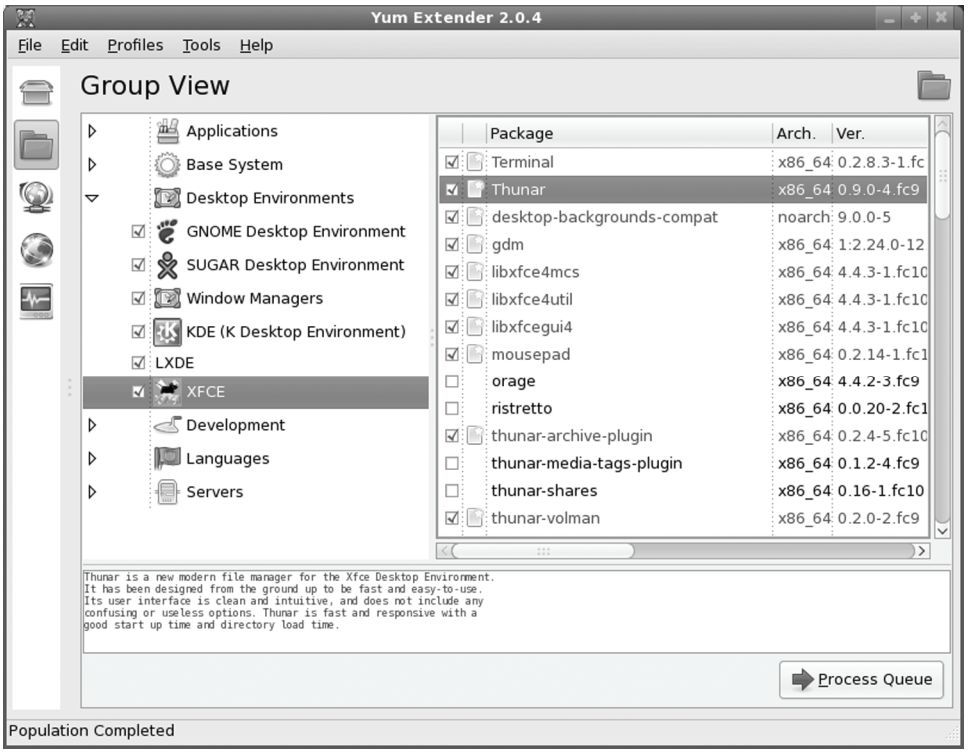


Figure 4-20: YUM Extender group view

Installing Packages with the yum command

You can also use the **yum** command in a terminal window or command line interface, to access Fedora repositories, downloading new software. To use the **yum** command, enter the **yum** command with the **install** option on a command line. The package will be detected, along with any dependent software, and you will be asked to confirm installation. The download and installation will be automatic. You can check the download repository online**,** as well as the repodata directories to display the available packages on a Web page. The page will include easy access features such as package groups and an index links based on the first letters of package names.

The following command installs Gnumeric:

```
yum install gnumeric
```

The following command installs KOffice from the Fedora repository:

```
yum install koffice
```

You can also remove packages, as well as search for packages and list packages by different criteria, such as those for update, those on the repository not yet installed, and those already installed. The following example lists all installed packages:

```
yum list installed
```

The available option shows uninstalled packages.

```
yum list available
```

To search for a particular package you can use the search option.

```
yum search xine
```

Tip: You can use Third Party Fedora YUM software repositories like http://rpmfusion.org to download additional software. Their configuration files will be in **/etc/yum.repos.d**.

Recovering packages with the yum command

Be sure to always check this dependency list and make sure a larger number of packages are not being marked for removal. In some cases, the entire GNOME desktop along with the GDM and PackageKit (Add/Remove Software) could be removed. If this is the case you could re-install GNOME, GDM, and PackageKit with **yum** commands in a terminal window or the command line interface. You might also have to reinstall graphics drivers. You can use **groupinstall** to install the entire GNOME desktop. The **grouplist** command displays groups.

```
yum groupinstall gnome-desktop
yum install gdm
yum install PackageKit
```

Vendor Video Driver support

You can obtain the Nvidia and ATI vendor graphics drivers from RPM Fusion (**nonfree** repository). These you can download and install with YUM. RPM Fusion, which now includes all the former Livna repository drivers, is the best repository for specialized kernel drivers and modules. Keep in mind though, that due to recent open sourcing of much of both the Nvidia and ATI vendor drivers, the Xorg open source versions are becoming almost as effective, especially for 2D display support. For normal usage you may not need vendor driver support.

The graphic drivers use two packages, one for the supporting software and another for the kernel. The kernel modules are specific to the kernel you are using. Once RPM Fusion is configured for PackageKit, a search on nvidia or fglrx will list the ATI and Nvidia modules for each kernel version. Each time you update to a new kernel, you will need a new graphics kernel module created specifically for that kernel. This will be automatically downloaded and installed for you as a dependent package when you update your kernel (the RPM Fusion nonfree repository will have to be active).

Note: The Nvidia and ATI vendor drives will be provided by the RPM Fusion repository as they become available for Fedora 10.

```
xorg-nvidia-x11-drv-nvidia-177
kmod-nvidia-177

xorg-x11-drv-fglrx
kmod-ati
```

The name of the ATI kernel module is **fglrx**, when available. The Xorg will be labeled **ati**, **xorg-x11-drv-ati**.

The Nvidia driver for GEForce 8 and above is **177**. There are also Nvidia driver package for **173** (Geforce 5-7), **legacy**, and **96xx** divers used for older Nvidia video cards. These cards also have their own Nvidia configuration tools. The Xorg open source drivers will be labeled **nv**, **xorg-x11-drv-nv**.

```
xorg-nvidia-x11-drv-nvidia-96xx
kmod-nvidia-96xx
xorg-nvidia-x11-drv-nvidia-legacy
kmod-nvidia-legacy
```

With RPM Fusion activated, you can download and install the graphics card driver using PackageKit, with Update System providing automatic updates. Use the Search box to search for either fglrx (ATI) or NVIDIA. You only need to use the initial unique term, **kmod-fgrlx**. YUM will detect the rest of the package name, selecting the package appropriate for your kernel. For Nvidia you would use **kmod-nvidia** in either PackageKit (Add/Remove Software), or in a **yum** command. YUM will detect and select **xorg-x11-drv-nvidia**, the Nvidia driver software.

Alternatively you could use a **yum** command entered in a terminal window.

```
yum install kmod-nvidia
```

Using repositories

A few repositories provide much of the software you will normally need. The main Fedora software repository will most likely contain the software you want. Always check this repository first, before trying a third-party repository. Some specialized applications like vendor-supplied graphics drivers, as well as third-party multimedia support, can be located at **http://rpmfusion.org**. Java applications are located at **http://jpackage.org**, though some are already in the Fedora repository. Together these repositories make up a set of software sites you can use to provide most of the functionality users expect from a desktop system. All are YUM compliant, with YUM configuration files designed for specific Fedora releases. The sites and their repository files are shown here.

http://download.fedoraproject.org	The Fedora Repository, mirror link: Fedora-compliant software. **fedora.repo**
http://download.fedora.redhat.com /pub/fedora/linux/releases	The Fedora Repository: Fedora-compliant software. **fedora.repo**
http://download1.rpmfusion.org	RPM Fusion: Repository for driver, multimedia, and other RPM packages, **rpmfusion-free.repo** and **rpmfusion-nonfree.repo**
http://jpackage.org	Java applications. **jpackage.repo**

To see what packages are available for a specific repository, you can use your Web browser to access its site. Fedora and RPM Fusion provide **repodata** directories for detailed listings.

To use YUM on a software repository, YUM has to be configured to access them. This is a simple matter of listing the site's URL, both its Web address and directory location. Configurations for repositories are placed in **repo** files located in the **/etc/yum.repos.d** directory on

your Linux system. The repo files for Fedora are already installed. The repo files for RPM Fusion have to be installed before you can access that repository with YUM. RPM Fusion provides an RPM packages or free and non-free collections, named **rpmfusion-free-release-stable.noarch.rpm** and rpmfusion-**nonfree-release-stable.noarch.rpm** on their Web site, that, when installed, automatically sets up the needed YUM configuration files. For RPM fusion these are **rpmfusion-free.repo** and **rpmfusion-nonfree.repo**, with corresponding update and development repo files.

Repository repo package files

To install packages, use PackageKit, yumex, or the **yum** command. All will automatically use the repo files to check your enabled repositories. One important exception to this rule is the initial install of the repository configuration files from third party repositories. These configuration files (repo files) are usually contained in their own RPM packages. You would download and install a repo package directly from the repository Web site using your Web browser. For example, the repo package for the RPM Fusion third party Fedora YUM repository **rpmfusion-free-release-stable.noarch.rpm**. You would download and install this package directly using a Web browser, being sure to select the prompted install option for the download. The PackageKit **gpk-install-file** tool will then be invoked to perform the installation.

Fedora Repository

The Fedora repository is already configured for use by YUM. To download any Fedora package, simply use the PackageKit program manager (System | Administration | Add/Remove Software), or enter the **yum** command with the **install** option on a command line. The package will be detected, along with any dependent software, and you will be asked to confirm installation. The download and installation will be automatic.

The first time you install a package from the Fedora repository, you will be prompted to install the Fedora GPG key, used to authenticate the packages. Just click Yes to install the key.

The Fedora repo file shown here lists several repository options. The name is Fedora, with **releaserver** and **basearch** used to determine the release and architecture parts of the name. The **mirrorlist** option is used instead of **baseurl**, which is commented out. Again, the **releaserver** is used to specify the release. The **gpgkey** used, RPM-GPG-KEY-fedora, is already installed in **/etc/pki/rpm-gpg** directory.

```
[fedora]
name=Fedora $releasever - $basearch
failovermethod=priority
#baseurl=http://download.fedoraproject.org/pub/fedora/linux/releases/$releasever/Everything/b
asearch/os/
mirrorlist=http://mirrors.fedoraproject.org/mirrorlist?repo=fedora-$releasever&arch=$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-$basearch
```

RPM Fusion

The RPM Fusion repository (**http://rpmfusion.org**) provides access to popular software for many software applications, including multimedia applications like MPlayer, as well as those not included with Fedora due to licensing issues. Several of the more popular packages include the

vendor ATI and NVIDIA graphics drivers. RPM Fusion specializes in configuring sometimes difficult drivers for compatibility with Fedora. For example, you can download the NVIDIA Linux driver directly from the NVIDIA Web site and try to install it on your Fedora system. But there can be complications and the driver could require additional configuration. A safer and more reliable approach is to simply install the RPM Fusion package for the NVIDIA driver. RPM Fusion provides a version of the driver that has already been configured for Fedora. RPM Fusion also provides ATI driver packages (**fglrx**).

To configure YUM on your system to access **http://rpmfusion.org**, just install the **rpmfusion-free-release** and **rpmfusion-nonfree-release** packages. These will install the **rpmfusion-free.repo** and **rpmfusion-nonfree.repo** configuration files in the **/etc/yum.repos.d** directory, as well as download the RPM Fusion GPG key. You can install the GPG manually or just wait until you first install a package from RPM Fusion, in which case YUM will install it for you, after prompting for your approval. Check the RPM Fusion configuration page at **http://rpmfusion.org** for details. The name of the package will be something like

```
rpmfusion-free-release
```

You can download and install the package directly with **rpm** command from within a terminal window. Login as root with **su** first or use the **sudo** command. The RPM Fusion site's configuration page provides a detailed example.

```
rpm -ivh http://download1.rpmfusion.org/free/fedora/rpmfusion-free-release-
stable.noarch.rpm
rpm -ivh http://download1.rpmfusion.org/nonfree/fedora/rpmfusion-nonfree-release-
stable.noarch.rpm
```

Alternatively, you can install directly from your browser by selecting Open With "Software Installer", or download the file, right-click on it, and select Open With "Software Installer". The PackageKit **gpk-install-file** tool will then be invoked to perform the installation. The package will be listed in the RPM Fusion category for the RPM Fusion Web display of the Fedora release.

You could also install it later with the **rpm** command as shown here:

```
rpm -ivh rpmfusion-free-release-stable.noarch.rpm
rpm -ivh rpmfusion-nonfree-release-stable.noarch.rpm
```

Once the repository is installed, all RPM Fusion packages will be accessible with YUM and integrated into PackageKit software listings.

The GPG keys are in included in the packages and installed automatically. They will be installed to the **/etc/pki/rpm-gpg** directory. They are named:

```
RPM-GPG-Key-rpmfusion-free-fedora
RPM-GPG-Key-rpmfusion-nonfree-fedora
```

To install manually, you could extract them with the Archive Manger and then use rpm with the **--import** command.

```
rpm --import RPM-GPG-Key-rpmfusion-free-fedora
```

To see what packages are available on RPM Fusion, use your Web browser to go to **http://download1.rpmfusion.org**. Here you will find a listing of directories for each supported

release. Selecting your release directory, you then choose your architecture, such as i386 (32 bit) or x86_64 (64 bit). Here you will see a simple file listing of all available packages.

Part of the **rpmfusion-free.repo** configuration file installed by the **rpmfusion-free-release** package is shown here. The configuration uses a mirror list instead of the baseurl site. The file also holds a testing configuration to let you access very new untested packages.

```
 [rpmfusion-free]
name=RPM Fusion for Fedora $releasever - Free
#baseurl=http://download1.rpmfusion.org/free/fedora/releases/$releasever/Everything/$basearch
/os/
mirrorlist=http://mirrors.rpmfusion.org/mirrorlist?repo=free-fedora$releasever&arch=$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-rpmfusion-free-fedora
```

The corresponding part of the **rpmfusion-nonfree.repo** configuration file installed by the **rpmfusion-nonfree-release** package is shown here.

```
[rpmfusion-nonfree]
name=RPM Fusion for Fedora $releasever - Nonfree
#baseurl=http://download1.rpmfusion.org/nonfree/fedora/releases/$releasever/Everything/$basea
rch/os/

mirrorlist=http://mirrors.rpmfusion.org/mirrorlist?repo=nonfree-
fedora$releasever&arch=$basearch

enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-rpmfusion-nonfree-fedora
```

Note: Livna, Freshrpms, and Dribble are now integrated with RPM Fusion.

Additional Java Applications: jpackage.org

Though Fedora already includes an extensive collection of Java software, including the OpenJDK and JRE, you can download additional Java applications from **http://jpackage.org**. Download and install the **jpackage.repo**. Place **jpackage.repo** in the **/etc/yum.repos.d** directory. Jpackage repo file configurations for different Linux distributions are displayed at **www.jpackage.org/yum.php**. Here you will find a listing for the latest Fedora distribution. The gpgkey is located online at **http://www.jpackage.org/jpackage.asc**.

YUM Configuration

YUM options are configured in the **/etc/yum.conf** file, and the **/etc/yum.repos.d** directory holds repository (repo) files that list accessible YUM repositories. The repository files have the extension **.repo**. Check the **yum.conf** Man page for a listing of the different YUM options along with entry formats. The **yum.conf** file consists of different segments separated by bracket-encased headers, the first of which is always the main segment. Segments for different YUM server repositories can follow, beginning with the repository label encased in brackets. On Fedora, however, these are currently placed in separate repository files in the **/etc/yum.repos.d** directory.

In addition to **yum.conf**, YUM also supports a **/etc/yum** directory that can hold additional configuration information. On Fedora, this directory has a **pluginconf.d** subdirectory with

configuration for YUM plugins. These include blacklist, whiteout, and refresh-packagekit. The blacklist and whiteout plugins are disabled, but the refresh-packagekit are enabled.

/etc/yum.conf

The **yum.conf** file will just have the main segment with settings for YUM options. There are general YUM options like **logfile**, which lists the location of YUM logs, and **distroverpkg**, which is used to determine which release to use. Packages will be downloaded to the directory specified with the **cachdir** option, in this case **/var/cache/yum**. You can elect to keep the downloaded packages or have them removed after they are installed. The **tolerant** option allows for package install errors, and the **retries** option specifies the number of times to try to access a package. Both **exactarch** and **obsoletes** apply to YUM updating procedures, invoked with the YUM **update** command. The **tolerant** option will allow package list errors such as those for already installed software, enabling installation of other packages in the list to continue. The **obsoletes** option is used for distribution level updates, and **exactarch** will only update packages in your specific architecture, such as i386 instead of i686. **gpgcheck** is a repository option that is set here globally for the **repo** files. It checks for GPG software signatures.

```
 [main]
cachedir=/var/cache/yum
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
tolerant=1
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
installonly limit=3

# PUT YOUR REPOS HERE OR IN separate files named file.repo
# in /etc/yum.repos.d
```

Repository Files: /etc/yum.repos.d

The repository entries in the repo files begin with a bracket-enclosed server ID, a single-word unique name. Repository-specific options govern the access of software repositories. These include **gpgcheck**, which checks for GPG software signatures, **gpgkey**, which specifies the location of the signature, and **mirrorlist**, which references a URL holding mirror sites. The repository-specific **name** option provides a name for the repository. The URL reference is then assigned to the **baseurl** option. There should be only one **baseurl** option, but you can list several URLs for it, each on its own line. With the **mirrorlist** option you can just list a URL for a list of mirrors, instead of listing each mirror separately in the **baseurl** option. The URL entries often make use of special variables, **releaserver** and **basearch**. The **releaserver** obtains the release information from the **distroverpkg** option set in the main segment. The **basearch** variable specifies the architecture you are using as determined by YUM, such as i386. The enabled option actually turns on access to the repository. By setting it to 0, you choose not to access a specific repository. The **gpgcheck** option specifies that you should perform a GPG authentication check to make sure the download is intact. The **enabled** option will enable a repository, allowing YUM to use it. You can set the enable bit to 0 or 1 to turn off or on access to the repository.

The **gpgkey** option provides an authentication check on the package to make sure you have downloaded the appropriate version. Sometimes downloads can be intercepted and viruses inserted. The GPG key check protects against such attacks. It can also check to make sure the download is not corrupt or incomplete. The Fedora public GPG key may already be installed on your system. If you have already used YUM, you will have already downloaded it. The Fedora GPG key allows you to access Fedora packages. The RPM Fusion free and non-free repositories use their own public keys, referenced with the **gpgkey** option in their repos files. The keys for all these repositories will be installed in **/etc/pki/rpmgpg** directory.

Creating Local YUM Repositories

For local networks where you may have several Fedora systems, each of which may need to update using YUM, you could set up a local repository instead of having each system update from Internet sites directly. In cases where local systems share a single Internet connection, this may significantly reduce download speeds. You can also control what packages can be installed. In effect, you download those packages on a YUM repository you want, and then create from those packages a local repository on one of your local systems. Your local systems them use the local repository to update install and update packages. You will have to manually keep the local repository updated. Use the **createrepo** command to create a repository from a directory holding the packages you want in it. Then it is a simple matter of providing a configuration file for it, specifying its location.

Managing YUM Caches

With the keepcache option enabled, YUM will keep its downloaded packages in the **/var/cache/yum** directory. Should you want to save or copy any particular packages, you can locate them there. Caching lets you easily uninstall and reinstall packages without having to download them again. The package is retained in the cache. If caching is disabled, then packages are automatically deleted after they are installed.

The size of your cache can increase rapidly, so you may want to clean it out on occasion. If you just want to deleted these packages, as they are already installed, you can use the clean packages option.

```
yum clean packages
```

YUM also maintains a list of package headers with information on all packages downloaded. The headers are used to update your system, showing what has already been installed. You can opt to remove the headers with the **clean headers** option.

If you want YUM to just access packages in the cache, you use the **-C** option. The following lists just packages in the cache.

```
yum -C list
```

Manually Installing Packages with rpm

If you are installing a package that is not part of a YUM repository, and you do not have access to the desktop or you prefer to work from the command line interface, you can use the **rpm** command to manage and install software packages (you can also open a terminal window by selecting the Terminal entry in the Applications | System Tools menu). In most cases you will not

need to use the **rpm** command. Most software now resides on YUM-supported, Fedora-compliant repositories. You would just use the **yum** command or the PackageKit (Add/Remove Software) front end, to install your package. YUM has the advantage of automatically installing any dependent packages, whereas the **rpm** command, though it will detect needed packages, will not install them. You will have to separately install any dependent packages in the correct order.

RPM has undergone a major upgrade with a complete rewrite with version 4.6. The actual command interface and options have not changed. The under package management has been made much more reliable and efficient. The rpm commands used remain the same.

For packages that are not part of any YUM-supported repository, such as custom-made packages and that have few or no dependent packages, you can use the **rpm** command directly. You could also use the **rpm** command to bypass YUM, forcing installation of a particular package instead of from YUM repositories (YUM's **localinstall** option will achieve the same purpose).

The **rpm** command performs installation, removal, and verification of software packages. Each software package is actually an RPM package, consisting of an archive of software files and information about how to install those files. Each archive resides as a single file with a name that ends with **.rpm**, indicating it is a software package that can be installed by the Red Hat Package Manager.

You can use the **rpm** command to either install or uninstall a package. The **rpm** command uses a set of options to determine what action to take. The **-i** option installs the specified software package, and the **-U** option updates a package. With an **-e** option, **rpm** uninstalls the package. A **q** placed before an **i** (**-qi**) queries the system to see if a software package is already installed and displays information about the software (**-qpi** queries an uninstalled package file). The **rpm** command with no options provides a complete list of **rpm** options. A set of commonly used options is shown in the following table.

Option	Action
-U	Updates package
-i	Installs package
-e	Removes package
-qi	Displays information for an installed package
-ql	Displays file list for installed package
-qpi	Displays information from an RPM package file (used for uninstalled packages)
-qpl	Displays file list from an RPM package file (used for uninstalled packages)
-K	Authenticates and performs integrity check on a package

The software package name is usually complex, including information about the version and release date in its name. All end with **.rpm**. In the next example, the user installs the **xvidcore** package using the **rpm** command. Notice that the full filename is entered. To list the full name, you can use the **ls** command with the first few characters and an asterisk. The following examples use the DivX xvidcore RPM packages.

```
ls xvid*
```

You can also use the * to match the remainder of the name, as in the following:

```
ls xvidccore-1*.rpm
```

In most cases, you are installing packages with the **-U** option, for update. Even if the package is not already installed, **-U** still installs it.

```
$ rpm -Uvh xvidcore--1.1.3-4.fc10.x86_64.rpm
```

When RPM performs an installation, it first checks for any dependent packages. These are other software packages with programs the application you are installing needs to use. If other dependent packages must be installed first, RPM cancels the installation and lists those packages. You can install those packages and then repeat the installation of the application. To determine if a package is already installed, use the **-qi** option with **rpm**. The **-q** stands for query. To obtain a list of all the files the package has installed, as well as the directories it installed to, use the **-ql** option. To query package files, add the **p** option. The **-qpi** option displays information about a package, and **-qpl** lists the files in it. The following example lists all the files in the **xvidcore** package:

```
$ rpm -qpl xvidcore-1.1.3-4.fc10.x86_64.rpm
```

To remove a software package from your system, first use **rpm -qi** to make sure it is actually installed, and then use the **-e** option to uninstall it. As with the **-qi** option, you needn't use the full name of the installed file. You only need the name of the application. In the next example, the user removes the DivX xvidcore from the system:

```
$ rpm -e xvidcore
```

Package Security Check

If you download a software package, you may want to check its integrity and authentication, making sure the package was not tampered with and that it was obtained from a valid source. YUM is configured to automatically perform this check on all software downloaded from your Fedora-compliant repositories. Each repository configuration file in the **/etc/yum.repos.d** directory will have its **gpgcheck** option set to 1. Should you want to turn off this check for a particular repository, you can set its **gpgcheck** option to 0.

To authenticate a package, you check its digital signature. Packages are signed with encrypted digital keys that can be decrypted using the public key provided by the author of the package. This public key has to first be downloaded and installed on the encryption tool used on your system. Fedora, along with most Linux distributions, uses the GNU Privacy Guard (GPG) encryption tool. To use a public key to authenticate an RPM package, you first have to install it in the RPM key database. For all RPM packages that are part of the Fedora distribution, you can use the Fedora public key, placed during installation in the **/etc/pki/rpm-gpg/RPM-GPG-KEY-fedora** file. Here you will also find the RPM GPG keys for all your configured repositories, including RPM Fusion. There will also be GPG keys for the Fedora test and rawhide repositories, **RPM-GPG-KEY-fedora-test** and **RPM-GPG-KEY-fedora-rawhide.**

The key has to be imported to the RPM database before you can check Fedora RPM packages. The first time you use PackageKit to install a package you will be prompted to import the GPG key. Once imported, you need not import it again. Alternatively, you can manually import the key as shown here:

```
rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-fedora
```

If you have downloaded an RPM package from another site, you can also download and install its public key, with which you can authenticate that package. For example there are public keys for both the RPM Fusion free and nonfree Fedora YUM repositories. These are included in the RPM Fusion YUM configuration files which you can download and install, like **rpmfusion-free-release-stable.noarch.rpm** for Livna. The keys will be automatically installed along with the configuration.

For RPM package files, once the public key for its repository or site is installed, you can check the package authentication using the **rpm** command with the **-K** option.

```
$ rpm -K xvidcore-1.1.3-4.fc10.x86_64.rpm
```

To see a list of all the keys you have imported, you can use the **-qa** option and match on the gpg-pubkey* pattern. Using **rpm** with the **-qi** option and the public key, you can display detailed information about the key. The following example shows the Fedora public key:

```
$ rpm -qa gpg-pubkey*
gpg-pubkey-4f2a6fd2-3f9d9d3b
gpg-pubkey-db42a60e-37ea5438
```

You can manually check just a package's integrity with the **rpm** command with the **-K** and the **--nosignature** options. A value called the MD5 digest measures the contents of a package. If the value is incorrect, the package has been tampered with. Some packages provide just digest values, allowing only integrity checks. In the next example, the user checks whether the **xvidcore** package has been tampered with. The **--nosignature** option says not to perform authentication, doing the integrity check only.

```
$ rpm -K --nosignature xvidcore-1.1.3-4.fc10.x86_64.rpm
```

Installing Source Code Applications

Many programs are available for Linux in source code format. These programs are stored in a compressed archive that you need to decompress and then extract. The resulting source code can then be configured, compiled, and installed on your system. The process has been simplified to the extent that it involves not much more than installing an RPM package. The following example shows the common method to extract, compile, and install software, in this case the freeciv program, a linux game. Always check the included README and INSTALL files that come with the source code to check the appropriate method for creating and installing that software.

Tip: Be sure that you have installed all development packages onto your system. Development packages contain the key components such as the compiler, GNOME and KDE headers and libraries, and preprocessors. You cannot compile source code software without them.

First you locate the software—in this example, from **www.linuxgames.com,** and then you download it to your system. freeciv is downloaded in a file named **- freeciv-2.1.8.tar.bz2**. Then decompress and extract the file either with the Archive Manager on the desktop, or with the **tar** command in a terminal window.

Extracting the Archive: Archive Manager (File Roller)

The easiest way to extract compressed archives is to use the Archive Manager (Applications | Accessories | Archive Manager) on GNOME (Archive Manager is the fileroller application). Either double-click the compressed archive file, or right-click and select Open With "Archive Manager". This displays the top-level contents of the archive, which you can browse if you wish, even reading text files like README and INSTALL files. You can also see what files will actually be installed. Use the button to navigate and double-click a directory to open it. Nothing is extracted at this point. To extract the archive, click Extract. You will be able to select what directory to extract to, the default will be a subdirectory in the current one (see Figure 4-21).

Archive Manager will let you actually read text files in the archive directly. Just select and double click the text file from its listing in the Archive manager window. Archive Manager will extract the file on the fly and display it in window (see Figure 4-22). The file is not actually extracted anywhere at this point.

You can also use Archive manager to create your own archives, creating an archive with the New button and then selecting files and folders for it with the Add File and Add Folder buttons.

Alternatively, on a command line, you can use the **`tar`** command to extract archives. To use the **`tar`** command, first open a terminal window (right-click on the desktop and select Open Terminal). At the prompt, enter the **`tar`** command with the **`xvjf`** options (**j** for **bz2** and **z** for **gz**), as shown here:

```
tar xvjf freeciv-2.1.8.tar.bz2
```

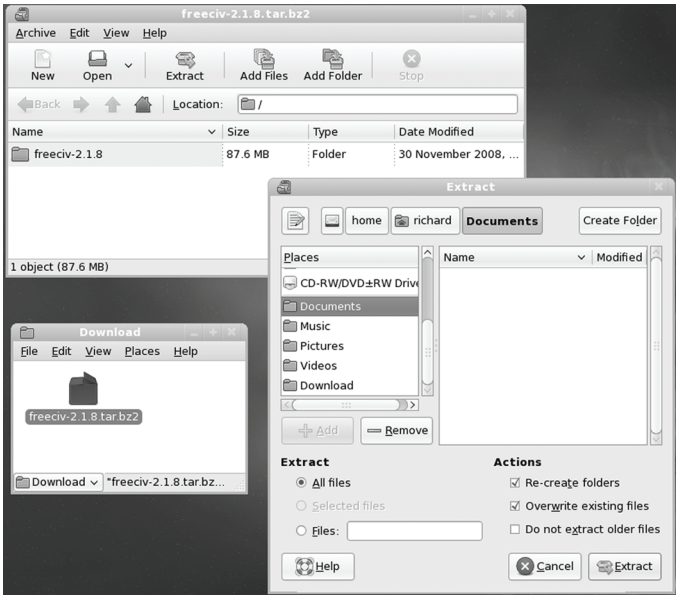


Figure 4-21: Archive Manager (File Roller) to exact software archive.

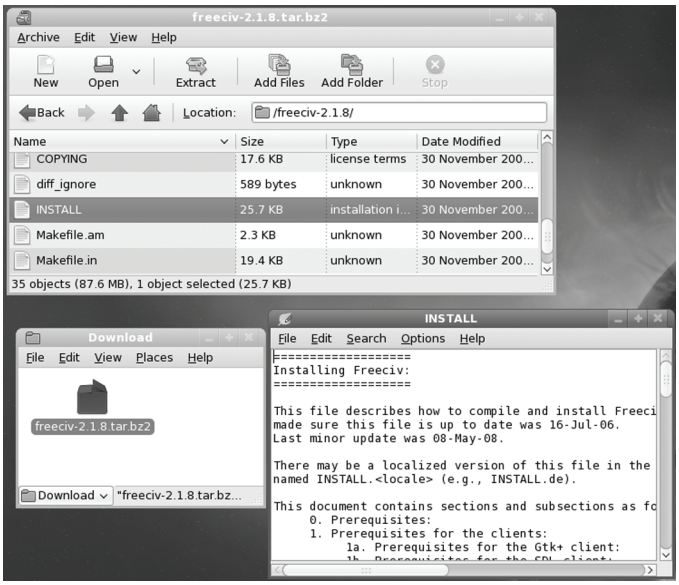


Figure 4-22: Archive Manager displaying archive text file.

Configure, Compile, and Install

Extracting the archive will create a directory with the name of the software, in this case **freeciv-2.1-4**. Once it is extracted, you have to configure, compile, and install the software. This usually needs to be done from a terminal window.

Change to the software directory with the **`cd`** command:

```
cd freeciv-2.1-4
```

Issue the command **`./configure`**. This generates a compiler configuration for your particular system (creates a custom Makefile).

```
./configure
```

Compile the software with the **`make`** command:

```
make
```

Then install the program with the **`make install`** command:

```
make install
```

Most KDE and GNOME supported software will also place an entry for the program in the appropriate menus—for example, a freeciv entry will be placed in the KDE Applications menu. You can then run freeciv from the menu entry. You could also open a terminal window and enter the program's name.

fedora

5. System Information

System Logs: /var/log and rsyslogd

The Linux Auditing System: auditd

Disk Usage Analyzer

Performance Analysis Tools and Processes

Information about your system is provided by system logs and performance analysis tools. These include the new Reliable and Extended Syslog logging application and a variety of performance tools like the GNOME System Monitor and the Disk Usage Analyzer.

System Logs: /var/log and rsyslogd

Various system logs for tasks performed on your system are stored in the **/var/log** directory. Here you can find logs for mail, news, and all other system operations, such as Web server logs. The **/var/log/messages** file is a log of all system tasks not covered by other logs. This usually includes startup tasks, such as loading drivers and mounting file systems. If a driver for a card failed to load at startup, you find an error message for it here. Logins are also recorded in this file, showing you who attempted to log in to what account. The **/var/log/maillog** file logs mail message transmissions and news transfers.

System logging is handled by the Reliable and Extended Syslog service, using the **rsyslogd** daemon. This replaces the older syslogd service. Configuration is now in the **/etc/rsyslog.conf** file. See www.rsyslog.com and the rsyslog documentation in **/usr/share/doc/rsyslog***. This documentation folder contains extensive Web page documentation on rsyslog.

GNOME Log Viewer

To view your logs you can use the GNOME Log Viewer, Applications | System Tools | System Logs. A side panel lists different logs. Selecting one will display the log on the panel to the right (see Figure 5-1).

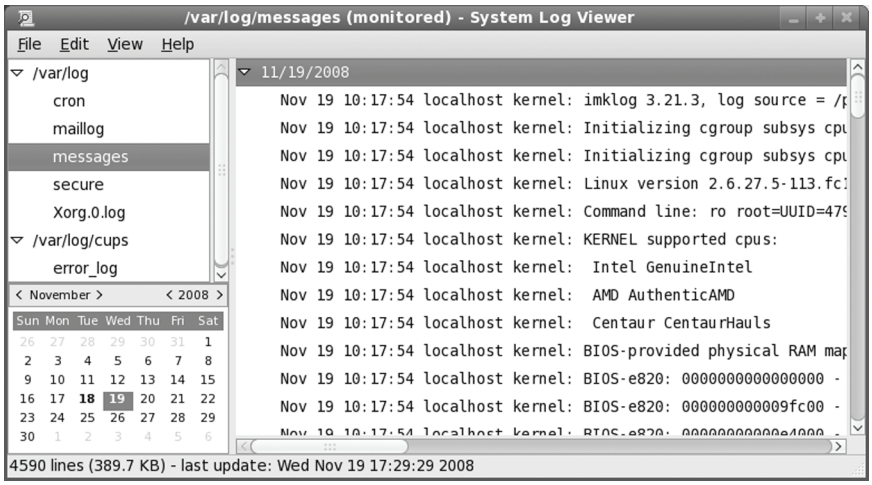


Figure 5-1: GNOME System Log Viewer

rsyslogd and /etc/rsyslog.conf

The **rsyslogd** daemon manages all the logs on your system, as well as coordinating with any of the logging operations of other systems on your network. Configuration information for r**syslogd** is held in the **/etc/rsyslog.conf** file, which contains the names and locations for your

system log files. Here you find entries for **/var/log/messages** and **/var/log/maillog**, among others. Whenever you make changes to the rsyslog.conf file, you need to restart the **rsyslogd** daemon using the following command (or use system-config-services, Server Settings | Services):

```
service syslog restart
```

Facilities	**Description**
auth-priv	Security/authorization messages (private)
cron	Clock daemon (cron and at) messages
daemon	Other system daemon messages
kern	Kernel messages
lpr	Line printer subsystem messages
mail	Mail subsystem messages
news	Usenet news subsystem messages
syslog	Syslog internal messages
user	Generic user-level messages
uucp	UUCP subsystem messages
local0 through **local7**	Reserved for local use
priorities	**Description**
debug	7, Debugging messages, lowest priority
info	6, Informational messages
notice	5, Notifications, normal, but significant, condition
warning	4, Warnings
err	3, Error messages
crit	2, Critical conditions
alert	1, Alerts, action must be taken immediately
emerg	0, Emergency messages, system is unusable
Operators	**Description**
*	Matches all facilities or priorities in a sector
=	Restrict to a specified priority
!	Exclude specified priority and higher ones
/	A file to save messages to
@	A host to send messages to
|	A FIFO pipe to send messages to

Table 5-1: rsyslogd Facilities, Priorities, and Operators

Entries in rsyslogd.conf

An entry in **rsyslog.conf** consists of two fields: a *selector* and an *action* (rsyslog uses the same selectors as the older syslog, maintaining compatibility). The selector is the kind of service to be logged, such as mail or news, and the action is the location where messages are to be placed. The action is usually a log file, but it can also be a remote host or a pipe to another program. The kind of service is referred to as a *facility*. The **rsyslogd** daemon has several terms it uses to specify certain kinds of service (see Table 5-1). A facility can be further qualified by a priority. A *priority* specifies the kind of message generated by the facility; **rsyslogd** uses several designated terms to indicate different priorities. A *selector* is constructed from both the facility and the priority, separated by a period. For example, to save error messages generated by mail systems, you use a sector consisting of the **mail** facility and the **err** priority, as shown here:

```
mail.err
```

To save these messages to the **/var/log/maillog** file, you specify that file as the action, giving you the following entry:

```
mail.err /var/log/maillog
```

The **rsyslogd** daemon also supports the use of * as a matching character to match either all the facilities or all the priorities in a sector: **cron.*** would match on all **cron** messages no matter what the priority, ***.err** would match on error messages from all the facilities, and ***.*** would match on all messages. The following example saves all mail messages to the **/var/log/maillog** file and all critical messages to the **/var/log/mycritical** file:

```
mail.* /var/log/maillog
*.crit /var/log/mycritical
```

Priorities

When you specify a priority for a facility, all messages with a higher priority are also included. The **err** priority also includes the **crit**, **alert**, and **emerg** priorities. If you just want to select the message for a specific priority, you qualify the priority with the **=** operator. For example, **mail.=err** will select only error messages, not **crit**, **alert**, or **emerg** messages. You can also restrict priorities with the **!** operator. This will eliminate messages with the specified priority and higher. For example, **mail.!crit** will exclude **crit** messages, as well as the higher **alert** and **emerg** messages. To specifically exclude all the messages for an entire facility, you use the **none** priority; for instance, **mail.none** excludes all mail messages. This is usually used when you're defining several sectors in the same entry.

You can list several priorities or facilities in a given sector by separating them with commas. You can also have several sectors in the same entry by separating them with semicolons. The first example saves to the **/var/log/messages** file all messages with **info** priority, excluding all mail and authentication messages (**authpriv**). The second saves all **crit** messages and higher for the **uucp** and **news** facilities to the **/var/log/spooler** file:

```
*.info;mail.none;news.none;authpriv.none /var/log/messages
uucp,news.crit /var/log/spooler
```

Actions and Users

In the action field, you can specify files, remote systems, users, or pipes. An action entry for a file must always begin with a / and specify its full pathname, such as **/var/log/messages**. To log messages to a remote host, you simply specify the hostname preceded by an @ sign. The following example saves all kernel messages on **rabbit.trek.com**:

```
kern.* @rabbit.trek.com
```

To send messages to users, you list their login names. The following example will send critical news messages to the consoles for the users **chris** and **aleina**:

```
news.=crit chris,aleina
```

You can also output messages to a named pipe (FIFO). The pipe entry for the action field begins with a **|**. The following example pipes kernel debug messages to the named pipe **|/usr/adm/debug**:

```
kern.=debug |/usr/adm/debug
```

An Example for /etc/rsyslog.conf

The default **/etc/rsyslog.conf** file for Fedora systems is shown here. Messages are logged to various files in the **/var/log** directory.

/etc/rsyslog.conf

```
# Log all kernel messages to the console.
kern.*                                 /dev/console
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;news.none;authpriv.none;cron.none     /var/log/messages

# The authpriv file has restricted access.
authpriv.*                          /var/log/secure
# Log all the mail messages in one place.
mail.*                                 /var/log/maillog
# Log cron stuff.
cron.*                                 /var/log/cron
# Everybody gets emergency messages
*.emerg                               *
# Save mail and news errors of level err and higher in a special file.
uucp,news.crit                 /var/log/spooler
# Save boot messages also to boot.log
local7.*                            /var/log/boot.log
```

The Linux Auditing System: auditd

The Linux Auditing System provides system call auditing. The auditing is performed by a server called **auditd**, with logs saved to the **/var/log/audit** directory. It is designed to compliment SELinux which saves its messages to the **auditd** log in **/var/log/audit/audit.log** file. The audit logging service provides specialize logging for services like SELinux. Logs are located at

/var/log/audit. To refine the auditing, you can create audit rules to check certain system calls like those generated by a specific user or group.

You can use **/etc/init.d/auditd** service script to start up and shutdown the **auditd** server. Use system-config-services or the **service** command to start and stop the server.

```
service auditd start
```

Configuration for auditd is located in both the **/etc/audit/auditd.conf** and the **/etc/sysconfig/auditd** files. Primary configuration is handled with **/etc/audit/auditd.conf** where options like the log file name, log format, the maximum size of log files, and actions to take when disk space diminishes. See the auditd.conf man page for a detailed description of all options. The **/etc/sysconfig/auditd** file sets server start up options and locale location like **en_US**.

The audit package includes the auditd server and three commands: **autrace**, **ausearch**, and **auditctl**. You use **ausearch** to query the audit logs. You can search by various ids, process, user, group, or event, as well as by filename or even time or date. Check the **ausearch** man page for a complete listing. autrace is a specialized tool that lets you trace a specific process. It operates similar to strace, recording the system calls and actions of a particular process.

You can control the behavior of the **auditd** server with the **auditctl** tool. With **auditctl** you can turn auditing on an off., check the status, and add audit rules for specific events. Check the **auditctl** man page for a detailed description.

Audit rules are organized into pre-determined lists with a specific set of actions for system calls. Currently there are three lists: task, entry, and exit, and three actions: never, always, and possible. When adding a rule, the list and action are paired, separated by a comma, as in:

```
exit,always
```

To add a rule you use the **-a** option. With the **-S** option you can specify a particular system call, and with the **-F** option specify a field. There are several possible fields you can use such as loginuid (user login id), pid (process id), and exit (system call exit value). For a field you specify a value, such as **longinuid=510** for the user with a user login id of 510. The following rule, as described in the documentation, checks all files opened by a particular user.

```
auditctl -a exit,always  -S open -F loginuid=510
```

Place rules you want loaded automatically in the **/etc/audit/auditd.rules**. The **sample.rules** file in **/usr/share/doc/auditd*** directory lists rule examples. You can also create a specific file of audit rules and use **auditctl** with the **-R** option to read the rules from it.

Disk Usage Analyzer

The disk usage analyzer lets you see how much disk space is used and available on all your mounted hard disk partitions (see Figure 5-2). It will also check all LVM and RAID arrays. Usage is shown in simple graph, letting you see how much overall space is available and where it is. When you san the file system (Scan Filesystem button in toolbar), disk usage for all your directories is analyzed and displayed in the left pane, and on a graph in the right pane. Passing your mouse over a section in the graph will display its directory name and disk usage. In the left hand listing, each files system is first shown with a graph for its usage, as well as its size and number of top-level directories and files. Then the directories are shown, along with their size and contents (files and directories). A menu above the chart lets you choose either a ring chart or a tree map

chart. The tree map chart shows the largest directories and their subdirectories, imaged according to size.

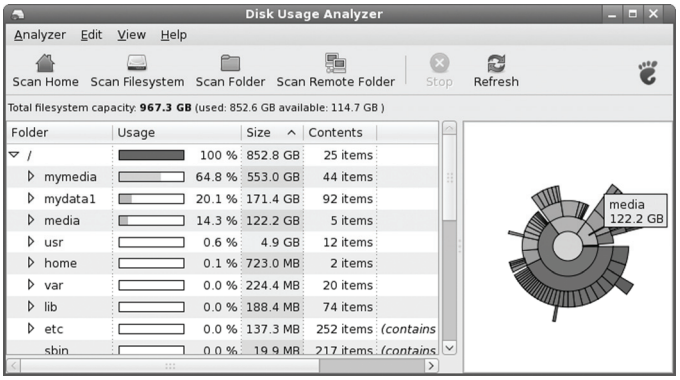


Figure 5-2: Disk Usage Analyzer, Applications | System Tools | Disk Usage Analyzer

Performance Analysis Tools and Processes

Linux treats each task performed on your system as a process, which is assigned a number and a name. You can examine these processes and even stop them. Fedora provides several tools for examining processes as well as your system performance. Easy monitoring is provided by the GNOME System Monitor (Applications | System Tools | System Monitor). Other tools are also available, such as GKrellM and KSysguard.

Performance Tool	Description
`vmstat`	Performance of system components
`top`	Listing of most CPU-intensive processes
`free`	Listing of free RAM memory
`sar`	System activity information
`iostat`	Disk usage
GNOME System Monitor	System monitor for processes and usage monitoring (System | Administration | System Monitor)
GKrellM	Stackable, flexible, and extensible monitoring tool
KDE Performance Monitor	KDE system monitor for processes and usage monitoring
Frysk	Monitoring tool for system processes
System Tap	Tool to analyze performance bottlenecks
Gnome Power Manager	Manage power efficiency features of your system
cpuspeed	Implement CPU speed reduction (AMD Cool and Quiet).

Table 5-2: Performance Tools

A number of utilities on your system provide detailed information on your processes, as well as other system information such as CPU and disk use (see Table 5-2). Although these tools were designed to be used on a shell command line, displaying output in text lines, several now have KDE and GNOME versions that provide a GUI interface for displaying results and managing processes.

GNOME System Monitor

Fedora provides the GNOME System Monitor for displaying system information and monitoring system processes, accessible from Applications | System Tools | System Monitor. There are four tabs: one for system information, one for processes, one for resources, and one for file systems (see Figure 5-3). The System tab shows the amount of memory, the kernel version, available disk space, and the type of CPU on your system.

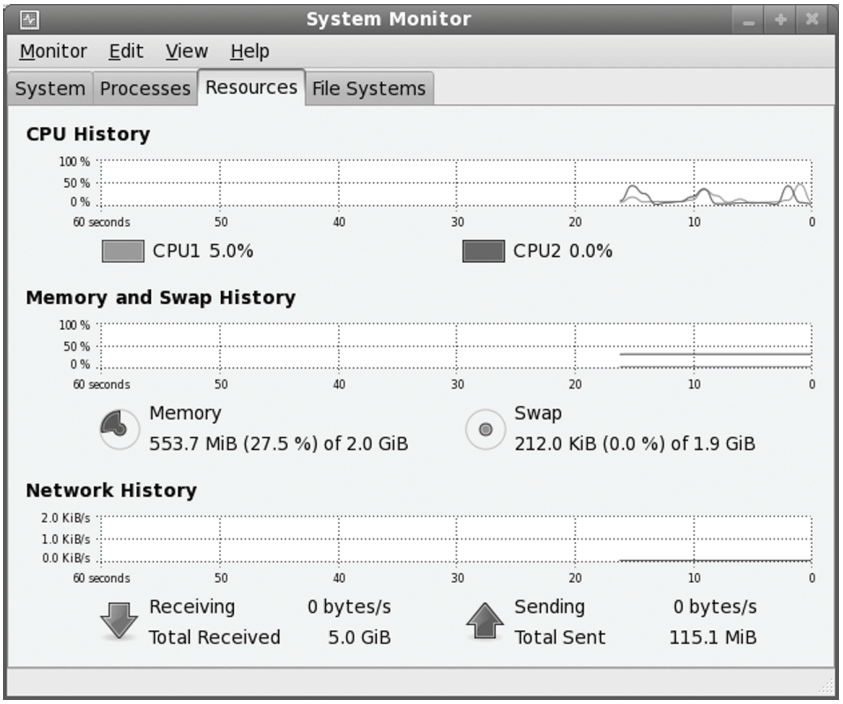


Figure 5-3: GNOME System Monitor

The Resources tab displays graphs for CPU, Memory and Swap memory, and Network usage. Your File Systems panel lists your file systems, where they are mounted, and their type, as well as the amount of disk space used and how much is free. The Processes tab lists your processes, letting you sort or search for processes. You can use field buttons to sort by name, process ID, user, and memory. The View pop-up menu lets you select all processes, just your own, or active processes. You can easily stop any process by selecting it and then clicking the End Process button. Right-clicking an item displays actions you can take on the process such as stopping or ending it. The Memory Maps display, selected from the View menu, shows information on virtual memory, inodes, and flags.

Managing Processes

Should you have to force a process or application to quit, you can use the Gnome System Monitor Processes tab to find, select, and stop the process. You should be sure of the process you want to stop. Ending a critical process could cripple your system. Application processes will bear the name of the application, and you can use those to force an application to quit. Ending processes manually is usually preformed for open ended operations that you are unable to stop normally. In Figure 5-4, the firefox application has been selected. Clicking the End Process button on the lower right will then force the Firefox Web browser to end (you can also right-click on the entry and select End process from the pop-up menu).

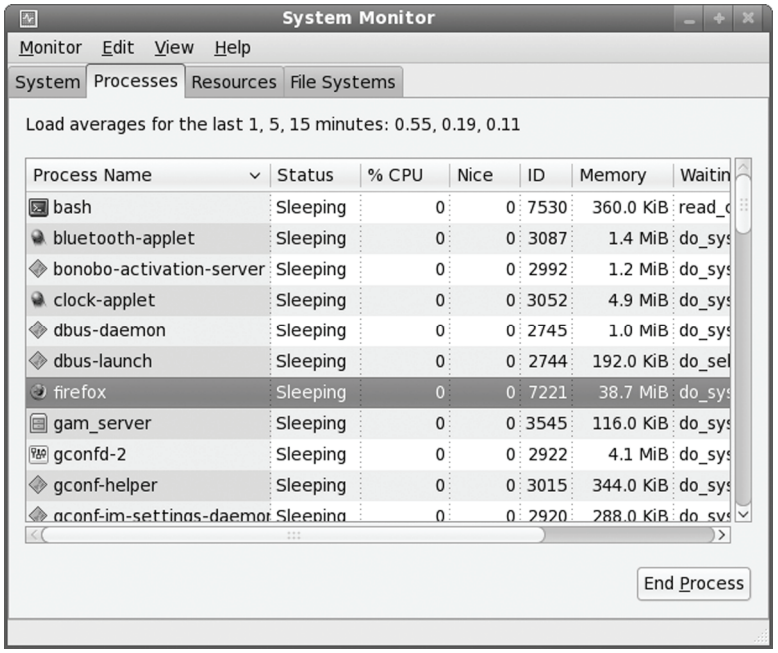


Figure 5-4: GNOME System Monitor, Processes tab

You can also use the kill command in a terminal window to end a process. The **kill** command takes as its argument a process number. Be sure you obtain the correct one. Use the **ps** command to display a process id. Entering in the incorrect process number could also cripple your system. The **ps** command with the **-C** option searches for a particular application name. The **-o pid=** will display only the process id, instead of the process id, time, application name, and tty. Once you have the process id, you can use the kill command with the process id as its argument to end the process.

```
$ ps -C firefox -o pid=
5555
$ kill 5555
```

One way to insure the correct number is to use the **ps** command to return the process number directly as an argument to a **kill** command. In the following example, an open ended

process was started to record a program from channel 12 from a digital video broadcast device, using the **getatsc** command.

```
getatsc -dvb 0 12 > my.ts
```

The process is then stopped by first executing the **ps** command to obtain the process id for the **getatsc** process (backquotes), and then using that process id in the **kill** command to end the process. The **-o pid=** option displays only the process id.

```
kill `ps -C getatsc -o pid=`
```

With the **-aux** option, you can list all processes. Piping the output to a **grep** command with a pattern enables you to search for a particular process. A pipe funnels the output of a preceding command as input to a following command. The following command lists all X Window System processes:

```
ps -aux | grep 'X'
```

vmstat, free, top, iostat, Xload, and sar

The **vmstat** command outputs a detailed listing indicating the performance of different system components, including CPU, memory, I/O, and swap operations. A report is issued as a line with fields for the different components. If you provide a time period as an argument, it repeats at the specified interval—usually a few seconds. The **top** command provides a listing of the processes on your system that are the most CPU intensive, showing what processes are using most of your resources. The listing is in real time and updated every few seconds. Commands are provided for changing a process's status, such as its priority.

The **free** command lists the amount of free RAM memory on your system, showing how much is used and how much is free, as well as what is used for buffers and swap memory. **Xload** is an X Window System tool showing the load, CPU, and memory, **iostat** displays your disk usage, and **sar** shows system activity information.

System Tap

System Tap is a diagnostic tool for providing information about complex system implementations. It essentially analyzes performance bottlenecks, letting you hone in on where a problem could be located. System Tap relies on Kprobes (Kernel Dynamic Probes) which allows kernel modules to set up simple probes.

Frysk

Frysk is a specialized complex monitoring tool for system processes. With Frysk you can set up very specific monitoring tasks, focusing on particular applications and selecting from a set of observer processes to provide information about exit notification, system calls, and execution. You can also create your own customized observers for processes. Find out more about Frysk at **http://sourceware.redhat.com/frysk**. Check **http://sourceware.org/frysk/javadoc/private/overview-summary.html for an overview**.

Gnome Power Manager

For power management, Fedora uses the GNOME Power Manager, gnome-power-manager, which makes use of ACPI support provided by a computer to manage power use. The GNOME Power manager is configured with the Power Management Preferences window (gnome-power-preferences), accessible from System | Preferences | System | Power Management. Power management can be used to configure both a desktop and a laptop. For a laptop there are only two panels, On AC Power and General.

The Gnome Power Manager is designed to take full advantage of the efficiency features available on both laptops and desktops. It supports tasks like reducing CPU frequency, dimming the display, shutting down unused hard drives, and automatic shutdown or suspension. See **http://projects.gnome.org/gnome-power-manager/index.html** for a detailed description. The Gnome Power Manger is integrated with HAL (Hardware Abstraction Layer) and Dbus to detect hardware states and issue hardware notifications. Hardware notifications are issued using notification icons for devices, such as the battery icon. The notification icons are located on the panel. A tooltip on the battery icon will show how much time you have left.

On the On AC Power window you have two sleep options, one for the computer and one for the screen. You can put each to sleep after a specified interval of inactivity. On the General panel you set desktop features like actions to take when pressing the power button or whether to display the power icon (see Figure 5-3). The AC power icon will show a plug image for desktops and a battery for laptops. The icon is displayed on the top panel.

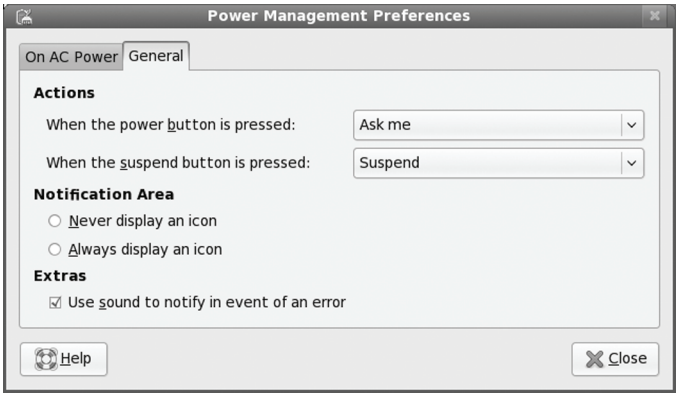


Figure 5-3: Power Management Preferences

On a Laptop, the battery icon displayed on the panel will show how much power you have left, as well as when the battery become critical. It will also indicate an AC connection, as well as when the battery has recharged.

To see how your laptop or desktop is performing with power, you can use Power statistics, Applications | System Tools.

Features like Cool and Quit for Athlon CPUs and Intel frequency controls are handled separately by the cpuspeed service. CPU frequency reporting tools are provided by cpufreq and gkrellm-freq packages. The **cpufreq-utils** package installs two applications, **cpu-info** and **cpu-set**

which will require CPU frequency drivers set in the kernel configuration, and can be compiled as modules.

GKrellM

GKrellM is a GTK-based set of small stackable monitors for various system, network, and device operations. A title bar at the top of the stack will display the host name of your system. By default, GKrellM will display the host name, system time, CPU load, process chart, disk access, network devices like **eth0**, memory use, and a mail check. You can change the chart display of a monitor, like its height, by right-clicking it to show a display options panel.

Each monitor will have a title bar, showing, for instance, CPU for CPU load, Disk for disk access, and Mem for memory. To configure the monitor, right-click its title bar. This will display the configuration panels for that task. For example, the Disk configuration will let you choose particular hard disks and partitions to monitor. The full configuration window will be displayed, showing a sidebar with configuration menus, with the built-in menu expanded to the selected monitor. The GKrellM monitors and configuration window.

See the **gkrellm** Man page for a detailed description of all monitor configuration options. The GKrellM site, **www.gkrellm.net**, offers resources for documentation, program support, and themes. Fedora comes with GKrellM installed with the built-in plug-ins and with the WiFi plug-in. Added plug-ins and themes can be downloaded from Fedora repository.

GKrellM Configuration

You can open the configuration window directly by clicking any monitor and pressing F1. Alternatively, you can open the main menu and select the Configuration entry. To open the main menu, either press F2 or right-click the top monitor. You can use this same menu to move through themes or to quit GKrellM. The configuration window shows a sidebar of configuration entries such as General, Builtins, Plugins, and Themes. Panels to the right let you set the configuration options. The General panel shows global options such as displaying the host name, the overall window size, and window priority.

The Builtins entry will expand to show a list of all the monitors that GKrellM can display. This list is extensive, including monitors as varied as fan and heat sensors, clock, CPU load, Internet connections, mail notification, and battery use.

The Plugins entry will expand to the list of installed plug-ins. To see available plug-ins, click the Plugin entry directly. A list of available plug-ins will be shown with check boxes. To install a plug-in, click its check box. Red Hat and Fedora may initially list just the WiFi monitor. As you install other plug-in packages, more will be listed. When installed, a plug-in will appear in the expanded plug-in menu. You can select a plug-in there to display its configuration panels.

Plug-ins for system-wide use are located in the **/usr/lib/gkrellm2/plugins** directory, and themes are located at **/usr/share/gkrellm2/themes**. User GKrellM configuration information and support files are kept in the **.gkrellm2** directory. Here you will find subdirectories for themes and user plug-ins. The **user_config** file holds user configuration settings, listing a monitor, its options, and its settings on each line.

GKrellM Server

You can use GKrellM to monitor hosts remotely using the GKrellM server, **gkrellmd**. You run the server on the system you wish to monitor, letting it allow remote systems to use **gkrellm** clients to gather and display its monitoring statistics. To run **gkrellm** as a client to gather and display information from another system running a **gkrellmd** server, you use the **-s** option and the server's host name. The server has to be configured to allow that remote host to connect. In the next example, a remote host connects to a **gkrellmd** server running on **turtle.mytrek.com** to display information about that the turtle host:

```
gkrellm -s turtle.mytrek.com
```

GKrellM is a service managed by the **/etc/init.d/gkrellmd** script. You can start and stop it with system-config-services or with the **service** command.

```
service gkrellmd start
```

Configuration for the GKrellM server is handled by the **/etd/gkrellmd.conf** configuration file. Here you can specify the hosts to monitor as well as global options such as the frequency of updates, the port to listen on, and the maximum number of simultaneous clients. Options are documented in detail. Check the Man page for **gkrellmd** for a complete listing.

KDE Task Manager and Performance Monitor (KSysguard)

Fedora also provides the KDE Task Manager and Performance Monitor, KSysguard, accessible from the System Tools menu as KDE System Guard only on the KDE desktop. This tool allows you to monitor the performance of your own system as well as remote systems. KSysguard can provide simple values or detailed tables for various parameters. A System Load panel provides graphical information about CPU and memory usage, and a Process Table list current processes using a tree format to show dependencies. You can design your own monitoring panels with worksheets, showing different types of values you want to display and the form you want to display them in, like a bar graph or digital meter. The Sensor Browser pane is an expandable tree of sensors for information like CPU System Load or Memory's Used Memory. There is a top entry for each host you are connected to, including your own, localhost. To design your own monitor, create a worksheet and drag and drop a sensor onto it.

fedora

6. Managing Users

system-config-users

User Configuration Files

Passwords

Managing User Environments

Adding and Removing Users with useradd, usermod, and userdel

Managing Groups

Disk Quotas

Lightweight Directory Access Protocol

As a system administrator, you can manage the users of your system. You can add or remove users, as well as add and remove groups, and you can modify access rights and permissions for both users and groups. You can decide how new user accounts should be configured initially. You also have access to system shell configuration files you can use to configure all user shells.

User Configuration Files

Any utility to manage a user, such as the system-config-users, makes use of certain default files, called *configuration files,* and directories to set up the new account. A set of pathnames is used to locate these default files or to indicate where to create certain user directories. For example, **/etc/skel** holds initialization files for a new user. A new user's home directory is created in the **/home** directory. Table 6-1 has a list of the pathnames.

Directory and Files	Description
/home	Location of the user's own home directory.
/etc/skel	Holds the default initialization files for the login shell, such as **.bash_profile**, **.bashrc**, and **.bash_logout**. Includes many user setup directories and files such as **.kde** for KDE and **Desktop** for GNOME.
/etc/shells	Holds the login shells, such as BASH or TCSH.
/etc/passwd	Holds the password for a user.
/etc/group	Holds the group to which the user belongs.
/etc/shadow	Encrypted password file.
/etc/gshadow	Encrypted password file for groups.
/etc/login.defs	Default login definitions for users.

Table 6-1: Paths for User Configuration Files

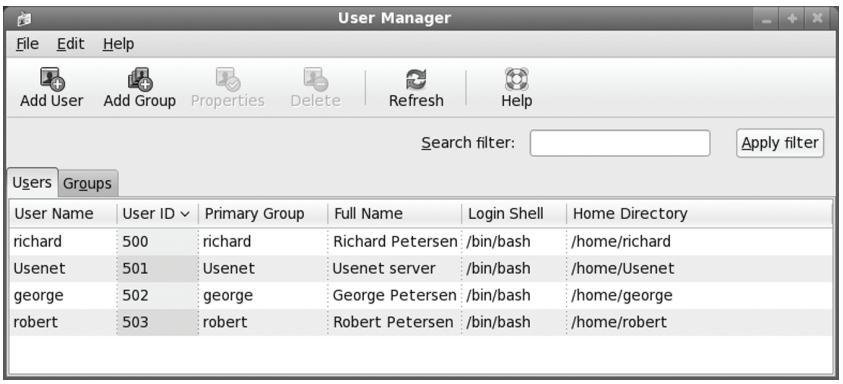


Figure 6-1: Users and Groups:system-config-users

system-config-users

Currently, the easiest and most effective way to add new users on Fedora is to use system-config-users, also known as the User Manager. You can access it from the GNOME Desktop menu, System | Administration menu | Users and Groups entry. The system-config-users window will display panels for listing both users and groups (see Figure 6-1). A button bar will list various tasks you can perform, including creating new users or groups, editing current ones (Edit), or deleting a selected user or group.

New Users

To create a new user, click Add User. This opens a window with entries for the username, password, and login shell, along with options to create a home directory and a new group for that user (see Figure 6-2).

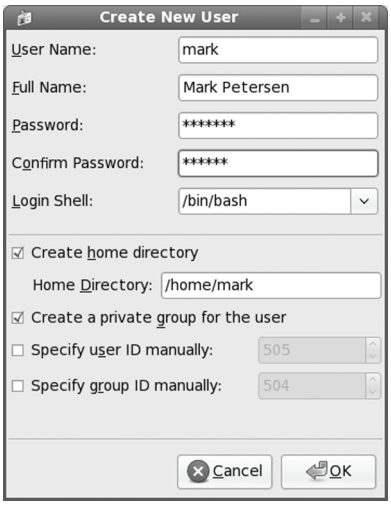


Figure 6-2: Users and Groups: Create New User

Once you have created a user, you can edit its properties to add or change features. Select the user's entry and click Properties. This displays a window with tabbed panels for User Data, Account Info, Password Info, and Groups (see Figure 6-3).

On the Groups panel, you can select the groups that the user belongs to, adding or removing group memberships (see Figure 6-3). The Accounts Info panel lets you set an expiration date for the user, as well as lock the local password. Password Info can enable password expiration, forcing users to change their passwords at certain intervals.

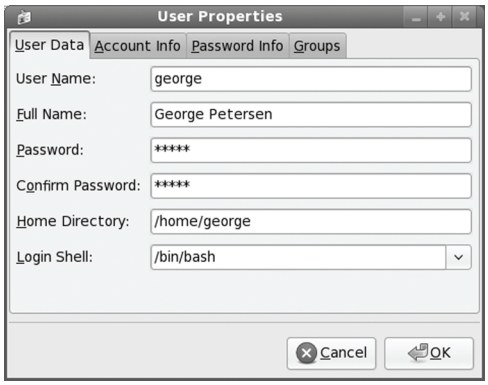


Figure 6-3: User Properties window: User Data

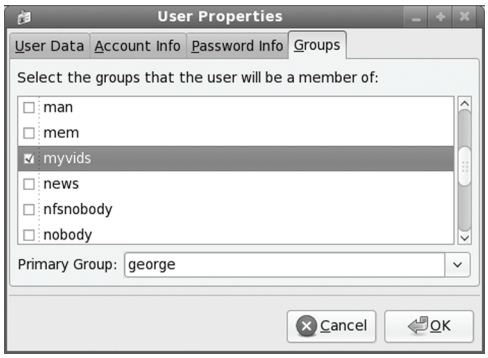


Figure 6-4: User Properties: Add groups to a user

Groups

To add a group, just click the Add Group button. This opens a small window where you can enter the group name. The new group will be listed in the Groups listing (see Figure 6-5).

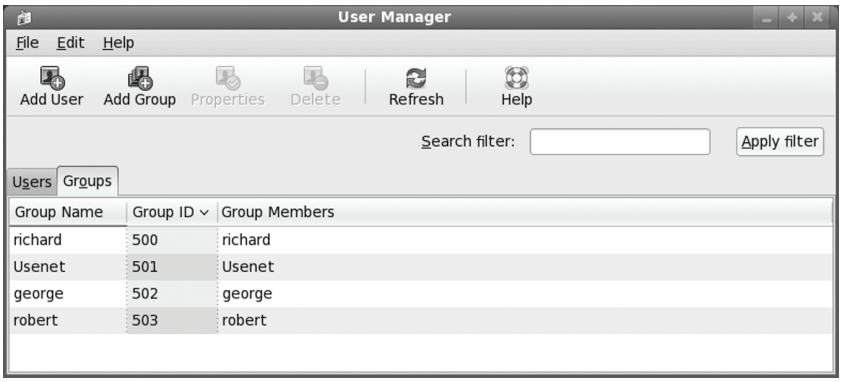


Figure 6-5: Users and Groups: Groups panel

To add users as members of the group, select the group's entry and click the Properties button. This opens a window with tabbed panels for Group Data and Group Users. The Group Users panel lists all current users with check boxes (see Figure 6-6). Click the check boxes for the users you want to be members of this group.

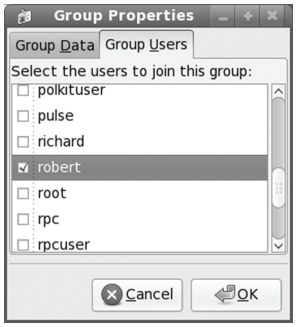


Figure 6-6: Group Properties: Group Users panel

If you want to remove a user as member, click the check box to remove its check. Click OK to effect your changes. If you want to remove a group, just select its entry in the Groups panel and then click the Delete button.

You can also add groups to a user by selecting the user in the Users panel, and opening their Properties window. Then select the Groups panel (see Figure 6-3). Select the groups you wan that user to belong to.

Passwords

A user gains access to an account by providing a correct login and password. The system maintains passwords in password files, along with login information like the user name and ID. Tools like the **`passwd`** command let users change their passwords by modifying these files; **/etc/passwd** is the file that traditionally held user passwords, though in encrypted form. However, all users are allowed to read the **/etc/passwd** file, which would have allowed access by users to the encrypted passwords. For better security, password entries are kept in the **/etc/shadow** file, which is restricted to the root user.

Tip: Every file is owned by a user who can controls access to it. System files are owned by the root user and accessible by the root only. Services like FTP are an exception to this rule. Though accessible by the root, a service's files are owned by their own special user. For example, FTP files are owned by an **ftp** user. This provides users with access to a service's files without also having root user access.

/etc/passwd

When you add a user, an entry for that user is made in the **/etc/passwd** file, commonly known as the *password file.* Each entry takes up one line that has several fields separated by colons. The fields are as follows:

- **Username** Login name of the user
- **Password** Encrypted password for the user's account
- **User ID** Unique number assigned by the system
- **Group ID** Number used to identify the group to which the user belongs
- **Comment** Any user information, such as the user's full name
- **Home directory** The user's home directory
- **Login shell** Shell to run when the user logs in; this is the default shell, usually /bin/bash

Depending on whether or not you are using shadow passwords, the password field (the second field) will be either an **x** or an encrypted form of the user's password. Fedora implements shadow passwords by default, so these entries should have an **x** for their passwords. The following is an example of an **/etc/passwd** entry. For such entries, you must use the **passwd** command to create a password. Notice also that user IDs in this particular system start at 500 and increment by one. With Fedora, the group given is not the generic User, but a group consisting uniquely of that user. For example, the **dylan** user belongs to a group named **Dylan**, not to the generic **User** group.

```
dylan:x:500:500:Dylan:/home/dylan:/bin/bash
chris:x:501:501:Chris:/home/chris:/bin/bash
```

Tip: If you turn off shadow password support, entries in your passwd file will display encrypted passwords. Because any user can read the **/etc/passwd** file, intruders can access and possibly crack the encrypted passwords.

Tip: Although it is technically possible to edit entries in the **/etc/passwd** file directly, it is not recommended. In particular, deleting an entry does not remove any other information, permissions, and data associated with a user, which opens a possible security breach whereby an intruder could take over the deleted user's ID or disk space.

/etc/shadow and /etc/gshadow

The **/etc/passwd** file is a simple text file and is vulnerable to security breaches. If anyone gains access to the **/etc/password** file, they might be able to decipher or crack the encrypted passwords through a brute-force crack. The shadow suite of applications implements a greater level of security. These include versions of **useradd**, **groupadd**, and their corresponding update and delete programs. Most other user configuration tools, including system-config-users, support shadow security measures. With shadow security, passwords are no longer kept in the **/etc/password** file. Instead, passwords are kept in a separate file called **/etc/shadow**. Access is restricted to the root user.

The following example shows the **/etc/passwd** entry for a user.

```
chris:x:501:501:Chris:/home/chris:/bin/bash
```

A corresponding password file, called **/etc/gshadow,** is also maintained for groups that require passwords. Fedora supports shadow passwords by default. You can manually specify whether you want to use shadow passwords with the system-config-authentication tool.

Password Tools

To change any particular field for a given user, you should use the user management tools provided, such as the **passwd** command, system-config-users, **adduser**, **usermod**, **useradd**, and **chage**, discussed in this chapter. The **passwd** command lets you change the password only. Other tools, such as system-config-users, not only make entries in the **/etc/passwd** file but also create the home directory for the user and install initialization files in the user's home directory.

These tools also let you control users' access to their accounts. You can set expiration dates for users or lock them out of their accounts. Users locked out of their accounts will have a their password in the **/etc/shadow** file prefixed by the invalid string, **!!**. Unlocking the account removes this prefix.

Tip: With the Fedora Authentication tool (system-config-authentication) you can enable and configure various authentication tools such as NIS and LDAP servers, as well as enabling shadow passwords, LDAP, and Kerberos authentication (accessible as Authentication on the System Settings menu and windows).

Users changing their own passwords

One common operation performed from the command line is to change a password. The easiest way to change your password on the GNOME desktop is to use the About Me utility (System | Preferences | Personal | About Me). click on the Change Password button. A dialog opens up in which you enter your current password, and then the new password.

Alternatively you can use the **passwd** command. If you are using GNOME or KDE you first have to open a terminal window (Applications | System Tools | Terminal). Then, at the shell prompt, enter the **passwd** command. The command prompts you for your current password. After entering your current password and pressing ENTER, you are then prompted for your new password. After entering the new password, you are asked to reenter it. This is to make sure you actually entered the password you intended to enter.

```
$ passwd
Old password:
New password:
Retype new password:
$
```

Tip: You can use the system-config-rootpassword tool (Root Password on System | Administration) to change the password for the root user.

Managing User Environments

Each time a user logs in, two profile scripts are executed, a system profile script that is the same for every user, and a user login profile script that can be customized to each user's needs. When the user logs out, a user logout script is run. In addition, each time a shell is generated, including the login shell, a user shell script is run. There are different kinds of scripts used for different shells. On Fedora and Red Hat Linux, the default shell commonly used is the BASH shell. As an alternative, users could use different shells such as TCSH or the Z shell, both installed with Fedora and Red Hat Linux.

Profile Scripts

For the BASH shell, each user has his or her own BASH login profile script named **.bash_profile** in the user's home directory. The system profile script is located in the **/etc** directory and named **profile** with no preceding period. The BASH shell user shell script is called **.bashrc**. The **.bashrc** file also runs the **/etc/bashrc** file to implement any global definitions such as the **PS1** and **TERM** variables. The **/etc/bashrc** file also executes any specialized initialization file in the **/etc/profile.d** directory, such as those used for KDE and GNOME. The **.bash_profile** file runs the **.bashrc** file, and through it, the **/etc/bashrc** file, implementing global definitions.

As a superuser, you can edit any of these profile or shell scripts and put in any commands you want executed for each user when that user logs in. For example, you may want to define a default path for commands, in case the user has not done so. Or you may want to notify the user of recent system news or account changes.

/etc/skel

When you first add a user to the system, you must provide the user with skeleton versions of their login, shell, and logout initialization files. For the BASH shell, these would be the **.bash_profile**, **.bashrc**, and **.bash_logout** files. The **useradd** command and other user management tools such as system-config-users add these files automatically, copying any files in the directory **/etc/skel** to the user's new home directory. The **/etc/skel** directory contains a skeleton initialization file for the **.bash_profile**, **.bashrc**, and **.bash_logout** files or, if you are using the TCSH shell as your login shell, the **.login**, **.tcshrc,** and **.logout** files. The **/etc/skel** directory also contains default files and directories for your desktops. These include a **.screenrc** file for the X Window System, a **.kde** directory for the KDE desktop, and a **Desktop** directory that contains default configuration files for the GNOME desktop.

As a superuser, you can configure the **.bash_profile** or **.bashrc** file in the **/etc/skel** directory any way you want. Usually, basic system variable assignments are included that define pathnames for commands and command aliases. The **PATH** and **BASH_ENV** variables are defined in **.bash_profile**. Once users have their own **.bash_profile** or **.bashrc** file, they can redefine variables or add new commands as they choose.

/etc/login.defs

System-wide values used by user and group creation utilities such as **useradd** and **usergroup** are kept in the **/etc/login.defs** file. Here you will find the range of possible user and group IDs listed. **UID_MIN** holds the minimum number for user IDs, and **UID_MAX** the maximum number. Various password options control password controls—such as **PASS_MIN_LEN**, which determines the minimum number of characters allowable in a password. Options such as **CREATE_HOME** can be set to tell user tools like **useradd** to create home directories for new accounts by default. Samples of these entries are shown here:

```
MAIL_DIR /var/spool/mail
PASS_MIN_LEN        5
CREATE_HOME yes
```

Controlling User Passwords

Once you have created a user account, you can control the user's access to it. Both the system-config-users and the **passwd** tool let you lock and unlock a user's account. You use the **passwd** command with the **-l** option to lock an account, invalidating its password, and you use the **-u** option to unlock it.

You can also force a user to change his or her password at given intervals by setting an expiration date for that password. Both system-config-users and the **chage** command let you specify an expiration limit for a user's password. A user could be required to change his or her password every month, every week, or at a given date. Once the password expires, the user will be prompted to enter a new one. You can issue a warning beforehand, telling the user how much time is left before the password expires. For account that you want to close, you can permanently expire a password. You can even shut down accounts that are inactive too long. In the next example, the password for the **chris** account will stay valid for only seven days. The **-M** option with the number of days sets the maximum time that a password can be valid.

```
chage -M 7  chris
```

Option	Description
-m	Minimum number of days a user must go before being able to change his password
-M	Maximum number of days a user can go without changing his password
-d	The last day the password was changed
-E	Specific expiration date for a password, date in format in yyyy-mm-dd or in commonly used format like mm/dd/yyyy
-I	Allowable account inactivity period (in days), after which password will expire
-W	Warning period, number of days before expiration when the user will be sent a warning message
-l	Display current password expiration controls

Table 6-2: Options for the chage Command

To set a particular date for the account to expire, use the **-E** option with the date specified mm/dd/yyyy.

```
chage -E 07/30/2008  chris
```

To find out what the current expiration settings are for a given account, use the **-l** option.

```
chage -l chris
```

You can also combine your options into one command,

```
chage -M 7 -E 07/30/2008  chris
```

A listing of the **chage** options appears in Table 6-2.

Adding and Removing Users with useradd, usermod, and userdel

Linux also provides the **useradd**, **usermod**, and **userdel** commands to manage user accounts. All these commands take in all their information as options on the command line. If an option is not specified, they use predetermined default values. These are command line operations. To use them on your desktop you first need to open a terminal window (right-click on desktop and select Open Terminal), and then enter the commands at the shell prompt.

If you are using a desktop interface with Red Hat and Fedora distributions, you should use the system-config-users to manage user accounts (see Chapter 6). You can access system-config-users from the System Settings menu under the Desktop menu. It is labeled simply as Users and Groups.

Options	Description
-d *dir*	Sets the home directory of the new user.
-D	Displays defaults for all settings. Can also be used to reset default settings for the home directory (**-b**), group (**-g**), shell (**-s**), expiration date (**-e**), and password expirations (**-f**).
-e *mm/dd/yy*	Sets an expiration date for the account (none, by default). Specified as month/day/year.
-f *days*	Sets the number of days an account remains active after its password expires.
-g *group*	Sets a group.
-m	Creates user's home directory, if it does not exist.
-m -k *skl-dir*	Sets the skeleton directory that holds skeleton files, such as **.profile** files, which are copied to the user's home directory automatically when it is created; the default is **/etc/skel**.
-M	Does not create user's home directory.
-p *password*	Supplies an encrypted password (crypt or MD5). With no argument, the account is immediately disabled.
-r	A Red Hat and Fedora–specific option that creates a system account (one whose user ID is lower than the minimum set in **logon.defs**). No home directory is created unless specified by **-m**.
-s *shell*	Sets the login shell of the new user. This is **/bin/bash** by default, the BASH shell.
-u *userid*	Sets the user ID of the new user. The default is the increment of the highest number used so far.

Table 6-3: Options for useradd and usermod

useradd

With the **useradd** command, you enter values as options on the command line, such as the name of a user, to create a user account. It then creates a new login and directory for that name using all the default features for a new account.

```
# useradd chris
```

The **useradd** utility first checks the **/etc/login.defs** file for default values for creating a new account. For those defaults not defined in the **/etc/login.defs** file, **useradd** supplies its own. You can display these defaults using the **useradd** command with the **-D** option. The default values include the group name, the user ID, the home directory, the **skel** directory, and the login shell. Values the user enters on the command line will override corresponding defaults. The group name is the name of the group in which the new account is placed. By default, this is **other**, which means the new account belongs to no group. The user ID is a number identifying the user account. The **skel** directory is the system directory that holds copies of initialization files. These initialization files are copied into the user's new home directory when it is created. The login shell is the pathname for the particular shell the user plans to use.

The **useradd** command has options that correspond to each default value. Table 6-3 holds a list of all the options you can use with the **useradd** command. You can use specific values in place of any of these defaults when creating a particular account. The login is inaccessible until you do. In the next example, the group name for the **chris** account is set to **intro1** and the user ID is set to 578:

```
# useradd chris -g intro1 -u 578
```

Once you add a new user login, you need to give the new login a password. Password entries are placed in the **/etc/passwd** and **/etc/shadow** files. Use the **passwd** command to create a new password for the user, as shown here. The password you enter will not appear on your screen. You will be prompted to repeat the password. A message will then be issued indicating that the password was successfully changed.

```
# passwd chris
Changing password for user chris
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
#
```

usermod

The **usermod** command enables you to change the values for any of these features. You can change the home directory or the user ID. You can even change the username for the account. The **usermod** command takes the same options as **useradd**, listed in Table 6-3.

userdel

When you want to remove a user from the system, you can use the **userdel** command to delete the user's login. With the **-r** option, the user's home directory will also be removed. In the next example, the user **chris** is removed from the system:

```
# userdel -r chris
```

Managing Groups

You can manage groups using either shell commands or window utilities like system-config-users. Groups are an effective way to manage access and permissions, letting you control several users with just their group name.

/etc/group and /etc/gshadow

The system file that holds group entries is called **/etc/group**. The file consists of group records, with one record per line and its fields separated by colons. A group record has four fields: a group name, a password, its ID, and the users who are part of this group. The Password field can be left blank. The fields for a group record are as follows:

- **Group name** The name of the group, which must be unique
- **Password** With shadow security implemented, this field is an x, with the password indicated in the /etc/gshadow file.
- **Group ID** The number assigned by the system to identify this group
- **Users** The list of users that belong to the group, separated by commas

Here is an example of an entry in an **/etc/group** file. The group is called **engines**, the password is managed by shadow security, the group ID is 100, and the users who are part of this group are **chris**, **robert**, **valerie**, and **aleina**:

```
engines:x:100:chris,robert,valerie,aleina
```

As in the case of the **/etc/passwd** file, it is best to change group entries using a group management utility like **groupmod**, **groupadd**, or system-config-users. All users have read access to the **/etc/group** file. With shadow security, secure group data such as passwords are kept in the **/etc/gshadow** file, to which only the root user has access.

User Private Groups

A new user can be assigned to a special group set up for just that user and given the user's name. Thus the new user **dylan** is given a default group also called **dylan**. The group **dylan** will also show up in the listing of groups. This method of assigning default user groups is called the User Private Group (UPG) scheme. UPG is currently used on Red Hat and Fedora systems. The supplementary groups are additional groups that the user may want to belong to. Traditionally, users were all assigned to one group named **users** that would subject all users to the group permission controls for the **users** group. With UPG, each user has its own group, with its own group permissions.

Group Directories

As with users, you can create a home directory for a group. To do so, you simply create a directory for the group in the **/home** directory and change its group to that of the group, along with allowing access by any member of the group. The following example creates a directory called **engines** and changes its group to that of the **engines** group:

```
mkdir /home/engines
chgrp engines /home/engines
```

Then the read, write, and execute permissions for the group level should be set with the **chmod** command, discussed later in this chapter:

```
chmod g+rwx /home/engines
```

Any member of the **engines** group can now access the **/home/engines** directory and any shared files placed therein. This directory becomes a shared directory for the group. You can, in fact, use the same procedure to make other shared directories at any location on the file system.

Files within the shared directory should also have their permissions set to allow access by other users in the group. When a user places a file in a shared directory, the user needs to set the permissions on that file to allow other members of the group to access it. A read permission will let others display it, write lets them change it, and execute lets them run it (used for scripts and programs). The following example first changes the group for the **mymodel** file to **engines**. Then it copies the **mymodel** file to the **/home/engines** directory and sets the group read and write permission for the **engines** group:

```
$ chgrp engines mymodel
$ cp mymodel /home/engines
$ chmod g+rw /home/engines/mymodel
```

Managing Groups with the system-config-users

You can add, remove, and modify any groups easily with system-config-users. First, access system-config-users by clicking the Users and Groups entry in the System Settings menu, listed in the Desktop menu. Then click the tabbed panel labeled Groups in the User Manager window. This will list all your current groups. There will be three fields for each entry: Group Name, Group ID, and Group Members.

To add a group, just click the Add Group button. This opens a small window where you can enter the group name. The new group will be listed in the Groups listing. To add users as members of the group, select the group's entry and click the Properties button. This opens a window with tabbed panels for Group Data and Group Users. The Group Users panel lists all current users with check boxes. Click the check boxes for the users you want to be members of this group. If you want to remove a user as member, click the check box to remove its check. Click OK to effect your changes. If you want to remove a group, just select its entry in the Groups panel and then click the Delete button.

Managing Groups Using groupadd, groupmod, and groupdel

You can also manage groups with the **groupadd**, **groupmod**, and **groupdel** commands. These command line operations let you quickly manage a group from a terminal window.

groupadd and groupdel

With the **groupadd** command, you can create new groups. When you add a group to the system, the system places the group's name in the **/etc/group** file and gives it a group ID number. If shadow security is in place, changes are made to the **/etc/gshadow** file. The **groupadd** command only creates the group category. You need to add users to the group individually. In the following example, the **groupadd** command creates the **engines** group:

```
groupadd engines
```

You can delete a group with the **groupdel** command. In the next example, the **engines** group is deleted:

```
groupdel engines
```

groupmod

You can change the name of a group or its ID using the **groupmod** command. Enter **groupmod -g** with the new ID number and the group name. To change the name of a group, you use the **-n** option. Enter **groupmod -n** with the new name of the group, followed by the current name. In the next example, the **engines** group has its name changed to **trains**:

```
groupmod -n trains engines
```

Disk Quotas

You can use disk quotas to control how much disk space a particular user makes use of on your system. On your Linux system, unused disk space is held as a common resource that each user can access as they need it. As users create more files, they take the space they need from the pool of available disk space. In this sense, all the users are sharing a single resource of unused disk space. However, if one user were to use up all the remaining disk space, none of the other users would be able to create files or even run programs. To counter this problem, you can create disk quotas on particular users, limiting the amount of available disk space they can use.

Quota Tools

Quota checks can be implemented on the file system of a hard disk partition mounted on your system. The quotas are enabled using the **quotacheck** and **quotaon** programs. They are executed in the **/etc/rc.d/rc.sysinit** script, which is run whenever you start up your system. Each partition needs to be mounted with the quota options, **usrquota** or **grpquota**. **usrquota** enables quota controls for users, and **grpquota** works for groups. These options are usually placed in the mount entry in the **/etc/fstab** file for a particular partition (see Chapter 15). For example, to mount the **/dev/sda4** hard disk partition mounted to the **/home** directory with support for user and group quotas, you would require a mount entry like the following:

```
/dev/sda4 /home ext2 defaults,usrquota,grpquota 1 1
```

You also need to create **quota.user** and **quota.group** files for each partition for which you enable quotas. These are the quota databases used to hold the quota information for each user and group. You can create these files by running the **quotacheck** command with the **-a** option or the device name of the file system where you want to enable quotas. The following example creates the quota database on the sda1 hard disk partition:

```
quotacheck -a  /dev/sda1
```

edquota

You can set disk quotas using the **edquota** command. With it, you can access the quota record for a particular user and group, which is maintained in the disk quota database. You can also set default quotas that will be applied to any user or group on the file system for which quotas have not been set. **edquota** will open the record in your default editor, and you can use your editor to make any changes. To open the record for a particular user, use the **-u** option and the username as

an argument for **edquota** (see Table 6-4). The following example opens the disk quota record for the user **larisa**:

```
edquota –u larisa
```

The limit you set for a quota can be hard or soft. A hard limit will deny a user the ability to exceed his or her quota, whereas a soft limit will just issue a warning. For the soft limit, you can designate a grace period during which time the user has the chance to reduce their disk space below the limit. If the disk space still exceeds the limit after the grace period expires, the user can be denied access to their account. For example, a soft limit is typically 75 megabytes, whereas the hard limit could be 100 megabytes. Users who exceed their soft limit could have a 48-hour grace period.

edquota Option	Description
-u	Edits the user quota. This is the default.
-g	Edits the group quota.
-p	Duplicates the quotas of the typical user specified. This is the normal mechanism used to initialize quotas for groups of users.
-t	Edits the soft time limits for each file system.

Table 6-4: Options for edquota

The quota record begins with the hard disk device name and the blocks of memory and inodes in use. The Limits segments have parameters for soft and hard limits. If these entries are 0, there are no limits in place. You can set both hard and soft limits, using the hard limit as a firm restriction. Blocks in Linux are currently about 1,000 bytes. The inodes are used by files to hold information about the memory blocks making up a file. To set the time limit for a soft limit, use the **edquota** command with the **-t** option. The following example displays the quota record for **larisa**:

```
Quotas for user larisa:
/dev/sda3: blocks in use: 9000, limits (soft = 40000, hard = 60000)
 inodes in use: 321, limits (soft = 0, hard = 0)
```

quotacheck, quotaon, and quotaoff

The quota records are maintained in the quota database for that partition. Each partition that has quotas enabled has its own quota database. You can check the validity of your quota database with the **quotacheck** command. You can turn quotas on and off using the **quotaon** and **quotaoff** commands. When you start up your system, **quotacheck** is run to check the quota databases, and then **quotaon** is run to turn on quotas.

repquota and quota

As the system administrator, you can use the **repquota** command to generate a summary of disk usage for a specified file system, checking to see what users are approaching or exceeding quota limits. **repquota** takes as its argument the file system to check; the **-a** option checks all file systems.

```
repquota /dev/sda1
```

Individual users can use the **quota** command to check their memory use and how much disk space they have left in their quota (see Table 6-5).

quota Option	Description
-g	Prints group quotas for the group of which the user is a member.
-u	Prints the user's quota.
-v	Displays quotas on file systems where no storage is allocated.
-q	Prints information on file systems where usage is over quota.

Table 6-5: Options for quota

Lightweight Directory Access Protocol

The Lightweight Directory Access Protocol (LDAP) is designed to implement network-accessible directories of information. In this context, the term directory is defined as a database of primarily read-only, simple, small, widely accessible, and quickly distributable information. It is not designed for transactions or updates. It is primarily used to provide information about users on a network, providing information about them such as their e-mail address or phone number. Such directories can also be used for authentication purposes, identifying that a certain user belongs to a specified network. You can find out more information on LDAP at **www.ldapman.org**. You can think of an LDAP directory for users as an Internet-accessible phone book, where anyone can look you up to find your e-mail address or other information. In fact, it may be more accurate to refer to such directories as databases. They are databases of user information, accessible over networks like the Internet. Normally, the users on a local network are spread across several different systems, and to obtain information about a user, you would have to know what system the user is on, and then query that system. With LDAP, user information for all users on a network is kept in the LDAP server. You only have to query the network's LDAP server to obtain information about a user. For example, Sendmail can use LDAP to look up user addresses. You can also use Firefox or Netscape to query LDAP.

Note: LDAP is a directory access protocol to an X.500 directory service, the OSI directory service.

Numerous LDAP packages are available on the Fedora respository that provide specific LDAP support. LDAP support packages include those for Java, Kerberos, Asterisk, Cyrus mail, DBmail, Mozilla SDK, nagios, NSS and PAM, Perl, ProFTPP, Ruby, Samba, and Wine

LDAP Clients and Servers

LDAP directories are implemented as clients and servers, where you use an LDAP client to access an LDAP server that manages the LDAP database. Fedora uses OpenLDAP, an open-source version of LDAP (you can find out more about OpenLDAP at **www.openldap.org**). This package includes an LDAP server (**slapd**), an LDAP replication server (**slurpd**), an LDAP client, and tools. **slurpd** is used to update other LDAP servers on your network, should you have more

than one. Once the LDAP server is installed, you can start, stop, and restart the LDAP server (**slapd**) with the **ldap** startup script or system-config-services:

```
service ldap restart
```

Tip: Red Hat and Fedora clients can enable LDAP services and select an LDAP server using the Fedora Authentication tool (system-config-authentication) accessible as the Authentication entry in the System Settings menu and window.

LDAP Configuration Files

All LDAP configuration files are kept in the **/etc/openldap** directory. These include **slapd.conf**, the LDAP server configuration file, and **ldap.conf**, the LDAP clients and tools configuration file. To enable the LDAP server, you have to manually edit the **slapd.conf** file, and change the domain value (dc) for the suffix and rootdn entries to your own network's domain address. This is the network that will be serviced by the LDAP server.

To enable LDAP clients and their tools, you have to specify the correct domain address in the **ldap.conf** file in the BASE option, along with the server's address in the HOST option (domain name or IP address). For clients, you can either edit the **ldap.conf** file directly or use the System Settings Authentication tool, clicking the Configure LDAP button on either the User Information or Authentication panel. Here you can enter your domain name and the LDAP server's address. See the **ldap.conf** Man entry for detailed descriptions of LDAP options.

The **/etc/ldap.conf** and **/etc/openldap/ldap.conf** files are not the same: **/etc/ldap.conf** is used to configure LDAP for the Nameservice Switch and PAM support, whereas **/etc/openldap/ldap.conf** is used for all LDAP clients.

Configuring the LDAP server: /etc/slapd.conf

You configure the LDAP server with the /etc/**slapd.conf** file. Here you will find entries for loading schemas and for specifying access controls, the database directory, and passwords. The file is commented in detail, with default settings for most options, although you will have to enter settings for several. First you need to specify your domain suffix and root domain manager. The default settings are shown here:

```
suffix          "dc=my-domain,dc=com"
rootdn          "cn=Manager,dc=my-domain,dc=com"
```

In this example, the **suffix** is changed to **mytrek**, for **mytrek.com**. The **rootdn** remains the same.

```
suffix          "dc=mytrek,dc=com"
rootdn          "cn=Manager,dc=mytrek,dc=com"
```

Next you will have to specify a password with **rootpw**. There are entries for both plain text and encrypted versions. Both are commented. Remove the comment for one. In the following example the plain text password option is used, "secret":

```
rootpw          secret
# rootpw        {crypt}ijFYNcSNctBYg
```

For an encrypted password, you can first create the encrypted version with **slappasswd**. This will generate a text encryption string for the password. Then copy the generated encrypted

string to the **rootpw** entry. On GNOME you can simply cut and paste from a terminal window to the **/etc/slapd.conf** file in Text Editor (Accessories). You can also redirect the encrypted string to a file and read it in later. SSHA encryption will be used by default.

```
# slappasswd
New password:
Re-enter new password:
{SSHA}0a+szaAwElK57Y8AoD5uMULSvLfCUfg5
```

The **rootpw** root password entry should then look like this:

```
rootpw      {SSHA}0a+szaAwElK57Y8AoD5uMULSvLfCUfg5
```

Use the password you entered at the **slappasswd** prompt to access your LDAP directory.

The configuration file also lists the schemas to be used. Schemas are included with the **include** directive.

```
include          /etc/openldap/schema/core.schema
include          /etc/openldap/schema/cosine.schema
include          /etc/openldap/schema/inetorgperson.schema
include          /etc/openldap/schema/nis.schema
```

LDAP directory database: ldif

A record (also known as entry) in an LDAP database begins with a name, known as a *distinguishing name,* followed by a set of attributes and their values. The distinguishing name uniquely identifies the record. For example, a name could be a username and the attribute would be the user's e-mail address, the address being the attribute's value. Allowable attributes are determined by schemas defined in the **/etc/openldap/schema** directory. This directory will hold various schema definition files, each with a **schema** extension. Some will be dependent on others, enhancing their supported classes and attributes. The basic core set of attributes are defined in the **core.schema** file. Here you will find definitions for attributes like country name and street address. Other schemas, like **inetorgperson.schema**, specify **core.schema** as a dependent schema, making its attributes available to the classes. The inetOrgPerson schema will also define its own attributes such as jpegPhoto for a person's photograph.

Schema Attributes and Classes

Attributes and classes are defined officially by RFC specifications that are listed with each attribute and class entry in the schema files. These are standardized definitions and should not be changed. Attributes are defined by an **attributetype** definition. Each is given a unique identifying number followed by a name by which it can be referenced. Fields include the attribute description (DESC), search features such as EQUALITY and SUBSTR, and the object identifier (SYNTAX). See the OpenLDAP administrative guide for a detailed description.

```
attributetype ( 2.5.4.9 NAME ( 'street' 'streetAddress' )
      DESC 'RFC2256: street address of this object'
      EQUALITY caseIgnoreMatch
      SUBSTR caseIgnoreSubstringsMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{128} )
```

A class defines the kind of database (directory) you can create. This will specify the kinds of attributes you can include in your records. Classes can be dependent, where one class becomes

and extension of another. The class most often used for LDAP databases is inetOrgPerson, defined in the **inetOrgPerson.schema** file. The term inetOrgPerson stands for Internet Organization Person, as many LDAP directories perform Internet tasks. The class is derived from the organizationalPerson class defined in **core.schema**, which includes the original attributes for commonly used fields like street address and name.

```
# inetOrgPerson
# The inetOrgPerson represents people who are associated with an
# organization in some way. It is a structural class and is derived
# from the organizationalPerson which is defined in X.521 [X521].
objectclass ( 2.16.840.1.113730.3.2.2
    NAME 'inetOrgPerson'
      DESC 'RFC2798: Internet Organizational Person'
    SUP organizationalPerson
    STRUCTURAL
      MAY (
            audio $ businessCategory $ carLicense $ departmentNumber $
            displayName $ employeeNumber $ employeeType $ givenName $
            homePhone $ homePostalAddress $ initials $ jpegPhoto $
            labeledURI $ mail $ manager $ mobile $ o $ pager $
            photo $ roomNumber $ secretary $ uid $ userCertificate $
            x500uniqueIdentifier $ preferredLanguage $
            userSMIMECertificate $ userPKCS12 )
      )
```

You can create your own classes, building on the standard ones already defined. You can also create your own attributes. But each attribute will require a unique object identifier (OID).

Distinguishing Names

Data in an LDAP directory is organized hierarchically, from general categories to specific data. An LDAP directory could be organized starting with countries, narrowing to states, then organizations and their subunits, and finally individuals. Commonly, LDAP directories are organized along the lines of Internet domains. In this format, the top category would be the domain name extension, for instance **.com** or **.ca**. The directory would then break down to the network (organization), units, and finally users.

This organization is used to help define distinguishing names that will identify the LDAP records. In a network-based organization, the top-level organization is defined by a domain component specified by the dcObject class, which includes the domainComponent (dc) attribute. Usually you define the network and extension as domain components to make up the top-level organization that becomes the distinguishing name for the database itself.

```
dc=mytrek, dc=com
```

Under the organization name is an organizational unit, such as users. These are defined as an organizationalUnitName (ou), which is part of the organizationalUnit class. The distinguishing name for the user's organizational unit would be.

```
ou=users, dc=mytrek, dc=com
```

Under the organizational unit you could then have individual users. Here the user name is defined with the commonName (cn) attribute, which is used in various classes, including Person,

which is part of organizationalPerson, which in turn is part of inetOrgPerson. The distinguishing name for the user **dylan** would then be:

```
cn=dylan,ou=users,dc=mytrek,dc=com
```

LDIF Entries

Database entries are placed in an LDAP Interchange Format (LDIF) file. This format provides a global standard that allows a database to be accessed by any LDAP-compliant client. An LDIF file is a simple text file with an **.ldif** extension, placed in the **/etc/openldap** directory. The entries for an LDIF record consist of a distinguishing name or attribute, followed by a colon, and its list of values. Each record begins with a distinguishing name to uniquely identify the record. Attributes then follow. You can think of the name as a record and the attributes as fields in that record. You end the record with an empty line.

Initially you create an LDIF file using any text editor. Then enter the records. In the following example, the **mytrek.ldif** LDIF file contains records for users on the network.

First you create records defining your organization and organization units. These distinguishing names will be used in user-level records. You will also have to specify a manager for the database, in this case simply Manager. Be sure to include the appropriate object classes. The organization uses both the dcObject (domain component object) and organization objects. The Manager uses organizationalRole, and users use the organizationalUnit. Within each record you can have attribute definitions, like the organization attribute, o, in the first record, which is set to MyTrek.

```
dn: dc=mytrek,dc=com
objectclass: dcobject
objectclass: organization
dc: mytrek
o: MyTrek

dn: cn=Manager,dc=mytrek,dc=com
cn: Manager
objectclass: organizationalRole

dn: ou=users,dc=mytrek,dc=com
objectclass: organizationalUnit
ou: users
```

Individual records then follow, such as the following for **dylan**. Here the object classes are organizationalPerson and inetOrgPerson. Attributes then follow, like common name (cn), user ID (uid), organization (o), surname (sn), and street.

```
dn: cn=dylan,ou=users,dc=mytrek,dc=com
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: dylan
uid: dylan
o: MyTrek
sn: shark
street: 77777 saturn ave
```

An example of an LDIF file is shown here. The organization is **mytrek.com**. There are two records, one for **dylan** and the other for **chris**:

mytrek.ldif

```
dn: dc=mytrek,dc=com
objectclass: dcobject
objectclass: organization
dc: mytrek
o: MyTrek

dn: cn=Manager,dc=mytrek,dc=com
cn: Manager
objectclass: organizationalRole

dn: ou=users,dc=mytrek,dc=com
objectclass: organizationalUnit
ou: users

dn: cn=dylan,ou=users,dc=mytrek,dc=com
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: dylan
uid: dylan
o: MyTrek
sn: shark
street: 77777 saturn ave

dn: cn=chris,ou=users,dc=mytrek,dc=com
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: chris
uid: chris
o: MyTrek
sn: dolphin
street: 99999 neptune way
```

Adding the Records

Once you have created your LDIF file, you can then use the **ldapadd** command to add the records to you LDAP directory. Use the **-D** option to specify the directory to add the records to, and the **-f** option to specify the LDIF file to read from. You could use **ldapadd** to enter fields directly. The **-x** option says to use simple password access, the **-W** will prompt for the password, and the **-D** option specifies the directory manager.

```
# ldapadd -x -D "cn=Manager,dc=mytrek,dc=com" -W -f mytrek.ldif

Enter LDAP Password:
adding new entry "dc=mytrek,dc=com"
adding new entry "cn=Manager,dc=mytrek,dc=com"
adding new entry "ou=users,dc=mytrek,dc=com"
adding new entry "cn=dylan,ou=users,dc=mytrek,dc=com"
adding new entry "cn=chris,ou=users,dc=mytrek,dc=com"
```

Be sure to restart the LDAP server to have your changes take effect.

Searching LDAP

Once you have added your records, you can use the **ldapsearch** command to search your LDAP directory. The **-x** and **-W** options provide simple password access, and the **-b** option will specify the LDAP database to use. Following the options are the attributes to search for, in this case the street attribute.

```
# ldapsearch -x -W -D 'cn=Manager,dc=mytrek,dc=com' -b 'dc=mytrek,dc=com' street
Enter LDAP Password:
# extended LDIF
#
# LDAPv3
# base <dc=mytrek,dc=com> with scope sub
# filter: (objectclass=*)
# requesting: street

# dylan, users, mytrek.com
dn: cn=dylan,ou=users,dc=mytrek,dc=com
street: 77777 saturn ave

# chris, users, mytrek.com
dn: cn=chris,ou=users,dc=mytrek,dc=com
street: 99999 neptune way

# search result
search: 2
result: 0 Success

# numResponses: 6
# numEntries: 5
```

If you want to see all the records listed in the database, you can use the same search command without any attributes.

LDAP Tools

To actually make or change entries in the LDAP database, you use the **ldapadd** and **ldapmodify** utilities (**openldap-clients** package). With **ldapdelete**, you can remove entries. Once you have created an LDAP database, you can then query it, through the LDAP server, with **ldapsearch**. For the LDAP server, you can create a text file of LDAP entries using the LDAP Data Interchange Format (LDIF). Such text files can then be read in all at once to the LDAP database using the **slapadd** tool. The **slapcat** tool extracts entries from the LDAP database and saves them in an LDIF file. To reindex additions and changes, you use the **slapindex** utility. See the LDAP Howto at the Linux documentation project for details on using and setting up LDAP databases such as address books (**www.tldp.org**).

Tip: You can enable and designate LDAP servers with the system-config-authentication tool (System | Administration | Authentication). You can also use the LDAP Browser/Editor or the GNOME Directory Administrator to manage and edit LDAP directories.

LDAP and the Name Service Switch Service

With the **libnss_ldap** library, LDAP can also be used in the Nameservice Switch (NSS) service along with NIS and system files for system database services like passwords and groups (**nss_ldap** package). Clients can easily enable LDAP for NSS by using the System Settings Authentication tool and selecting Enable LDAP Support in the User Information panel. You also need to make sure that the LDAP server is specified. You could also manually add **ldap** for entries in the **/etc/nsswitch.conf** file.

Tip: To better secure access to the LDAP server, you should encrypt your LDAP administrator's password. The LDAP administrator is specified in the **rootdn** entry, and its password in the **rootpw** entry. To create an encrypted password, use the **slappasswd** command. This prompts you for a password and displays its encrypted version. Copy that encrypted version in the **rootpw** entry.

Enabling LDAP on Thunderbird

In Thunderbird, open the address book, then select File | New, and choose the LDAPD directory. Here you can enter the LDAP server. This displays a panel where you can enter the address book name, the host name of LDAP server, the Base DN to search, and the port number, 389 on Fedora.

Fedora Directory Server

The Fedora Directory Server (FDS) is a Fedora implementation of the LDAP server, designed for enterprise level operations. Check the Fedora FDS site for detailed information. FDS is undergoing rapid development. It is advisable that you make use of the online documentation for up-to-date reference and instructions.

```
http://directory.fedoraproject.org/
```

Much of the documentation refers to earlier versions of FDS. Fedora 10 uses the latest, FDS 1.1 and higher. Check the FDS 1.1 install guide for installation details.

```
http://directory.fedoraproject.org/wiki/FDS_Setup
```

The Fedora 10 version operates like the Red Hat Directory Server 8.0 version. You can use the Red Hat documentation to learn how to install and operate your FDS. Some of the applicaton names will be different, like **fedora-idm-console** instead of **redhat-idm-console**.

```
http://www.redhat.com/docs/manuals/dir-server/install/8.0/index.html
```

Packages are prefixed with **fedors-ds** and include **fedora-ds-base**, **fedora-ds-admin**, and **fedora-ds-console**. The **fedora-ds** package is a meta-package that will install all the others.

```
fedora-ds-1.1.2-1.fc10
```

The FDS is comprised of three components: the Directory server, the Directory Server Console, and the Administration server along with its own administration console.

- **Directory Server** The main LDAP server. It includes command-line management tools.
- **Directory Server Console** The Web browser based interface that allows you to manage LDAP users and groups, as well as manage your server, making backups and setting up security.

- **Directory Administration Server** Administers the directory server, connecting to the Directory Server Console. It communicates with the Directory Server Console and performs operations on the Directory Server instances. Each directory server instance has it own directory administration server.
- **Directory Server Gateway** The Web browser based applications for the administration server. These include phonebook for end users providing simple search and basic self-service management, orgchart organization char viewer, and gateway for advanced search, creation, and editing of user entries (also supports templates), **fedora-ds-dwgw** package.

FDS has its own service script called **dirsrv** which is used to start and stop the directory server.

```
service dirsrv start
```

To first set up the FDS server, you run the setup-ds-admin.pl command. Be sure your system first has a fully qualified domain name (host and domain).

```
/user/sbin/setup-ds-admin.pl
```

You are given a series of prompts, with defaults provided. You can then choose to perform an express, typical, or custom configuration.

Once you have setup your FDS, you can access it with the console, **fedora-idm-console**.

```
/usr/bin/fedora-idm.console
```

Note: LDAP supports the Simple Authentication and Security Layer (SASL) for secure authentication with methods like MD5 and Kerberos.

fedora

7. Kernel Administration

The *kernel* is the operating system, performing core tasks such as managing memory and disk access, as well as interfacing with the hardware that makes up your system. For example, the kernel makes possible such standard Linux features as multitasking and multiuser support. It also handles communications with devices like your CD-ROM or hard disk. Users send requests for access to these devices through the kernel, which then handles the lower-level task of actually sending appropriate instructions to a device. Given the great variety of devices available, the kind of devices available to a Linux system will vary. When Linux is installed, the kernel is appropriately configured for available devices. However, if you add a rarely used or unsupported device, you may have to enable support for it in the kernel. This involves reconfiguring the existing kernel to support the new device through a procedure that is often referred to as *building* or *compiling the kernel.* In addition, new versions of the kernel are continuously made available that provide improved support for your devices, as well as support for new features and increased reliability for a smoother-running system. You can download, compile, and install these new versions on your system.

Kernel Versions

The version number for a Linux kernel consists of three segments: the major, minor, and revision numbers. The *major number* increments with major changes in the kernel. The *minor number* indicates stability. *Even numbers* are used for stable releases, whereas *odd numbers* are reserved for development releases, which may be unstable. The *revision number* refers to the corrected versions. As bugs are discovered and corrected, and as new features are introduced, new revisions of a kernel are released. For example, kernel **2.6.27.5** has a major number of 2 and a minor number of 6, with a revision number of 27, and a security/bug fix number of 5.

Distributions often add another number that refers to a specific set of patches applied to the kernel. For example, for Fedora, the kernel is version **2.6.27.5-117.fc10**, where 117 is the patch number and **fc10** refers to the fedora 10 distribution. The current kernel version is displayed on the System tab on the GNOME System Monitor (Applications | System Tools | System Monitor). You can also use an RPM query to learn what version is installed, as shown here:

```
rpm -q kernel
```

You could have more than one version of the kernel installed on your system. To see which one is running currently, you use the **uname** command with the **-r** option (the **-a** option provides more detailed information).

```
uname -r
```

New kernels are released on two different tracks, a stable track and a development track. Stable kernels have an even revision number, whereas development kernels use a odd number. The stable kernel would be 2.6, and its development kernel is 2.7. Often development kernels, though unstable, include support for the most recent hardware and software features. However, unless you are experimenting with kernel development, you should always install a stable version of the kernel.

New features first appear in the development versions. . New development versions will first appear as release candidates, which will have an **rc** in the name. If you're concerned about stability, you should wait for the stable version. A development kernel may have numerous revisions. The release candidate version would have a name like **2.6.28-rc7**.

The Linux kernel is being worked on constantly, and new versions are released when they are ready. Distributions may include different kernel versions. Fedora includes the most up-to-date stable kernel in its releases. Linux kernels are available at **www.kernel.org**. Also, RPM packages for a new kernel are often available at distribution update sites. One reason you may need to upgrade your kernel is to provide support for new hardware or for features not supported by your distribution's version. For example, you may need support for a new device not provided in your distribution's version of the kernel. Certain features may not be included in a distribution's version because they are considered experimental or a security risk.

Note: In many cases, you don't need to compile and install a new kernel just to add support for a new device. Kernels provide most device support in the form of loadable modules, of which only those needed are installed with the kernel. Most likely, your current kernel has the module you need; you simply have to compile it and install it.

Tip: Many modules can be separately compiled using sources provided by vendors, such as updated network device drivers. For these you only need the Kernel headers, which are already installed in the **/usr/lib/modules/***version***/source** directory, where *version* is an installed kernel version. In these cases, you do not have to install the full kernel source to add or modify modules.

References

You can learn more about the Linux kernel from **www.kernel.org**, the official repository for the current Linux kernels. The most current source code, as well as documentation, is there. The Fedora project also provides online documentation for installing and compiling the kernel on its systems, **http://fedoraproject.org/wiki/Building_a_custom_kernel**. Several Linux HOW-TOs also exist on the subject. The kernel source code software packages also include extensive documentation. Kernel source code files should always be installed in a local user directory (the **/usr/src/kernels** directory is used for library headers, not the kernel source). The source itself will be in a directory labeled **linux-***version,* where *version* is the kernel release, as in **linux-2.6.27**. In this directory, you can find a subdirectory named **/Documentation,** which contains an extensive set of files and directories documenting kernel features, modules, and commands. The following listing of kernel resources also contains more information:

- **www.kernel.org** The official Linux kernel Web site. All new kernels originate from here
- **www.linuxhq.com** Linux headquarters, kernel sources, and patches
- **http://kernelnewbies.org** Linux kernel sources and information
- **http://en.tldp.org/** Linux Documentation Project
- **http://fedoraproject.org/wiki/Building_a_custom_kernel** Fedora site for instructions for configurateur and building kernels on Fedora.

Kernel Tuning: Kernel Runtime Parameters

Several kernel features, such as IP forwarding or the maximum number of files, can be turned on or off without compiling and installing a new kernel or module. These tunable parameters are controlled by the files in **/proc/sys** directory. Parameters that you set are made in the

/etc/sysctl.conf file. Fedora installs this file with basic configuration entries such as those for IP forwarding and debugging control. You use the **sysctl** command directly. The **-p** option causes **sysctl** to read parameters from the **/etc/sysctl.conf** file (you can specify a different file). You can use the **-w** option to change specific parameters. You reference a parameter with its key. A key is the parameter name prefixed with its **proc** system categories (directories), such as **net.ipv4.ip_forward** for the **ip_forward** parameter located in **/proc/sys/net/ipv4/**. To display the value of a particular parameter, just use it's key. The **-a** option lists all available changeable parameters. In the next example, the user changes the domain name parameter, referencing it with the **kernel.domainname** key (the **domainname** command also sets the **kernel.domainname** parameter):

```
# sysctl -w kernel.domainname="mytrek.com"
```

The following example turns on IP forwarding:

```
# sysctl -w net.ipv4.ip_forward=1
```

If you use just the key, you display the parameter's current value:

```
# sysctl net.ipv4.ip_forward
 net.ipv4.ip_forward = 1
```

Installing a New Kernel Version

To install a new kernel, you need to download the software packages for that kernel to your system. You can install a new kernel either by downloading a binary version from your distribution's Web site and installing it or by downloading the source code, compiling the kernel, and then installing the resulting binary file along with the modules. For the binary version of the kernel is provided in an RPM package. You can install a new kernel, just as you would any other RPM software package.

The easiest way to install a new kernel on Fedora is to use Pirut. Pirut will automatically download and install a new kernel. The installation will create a GRUB entry so that when you boot, the new kernel will be listed as one of the options, usually the default.

If you want to download kernel RPM packages manually, keep in mind that the complete kernel installation usually includes a series of RPM packages, all beginning with the word *kernel*. There are also other packages you may need, which contain updated system configuration files used by the new kernel. You can use the packages already installed on your system as a guide. Use the **rpm** command with the **-qa** option to list all packages and then pipe that list through the **grep** command with the **kernel** pattern to display only the kernel packages:

```
rpm -qa | grep kernel
```

The kernel source code version is available for download from distribution FTP sites in the source directory and is included on distribution source code CD-ROM. You can also download the latest source directly from **www.kernel.org**. Wherever you download a kernel version from, it is always the same. The source code downloaded for a particular kernel version from a distribution site is the same as the one for **www.kernel.org**. Patches for that version can be applied to any distribution.

CPU Kernel Packages

Fedora and Red Hat provide different kernel packages optimized 32 bit and 64 bit CPUs. The one appropriate for your machine is included with the distribution. All the kernels include multi-processor support. The 32-bit distribution will include the x86 versions, and the 64-bit distribution will hold the x86_64 versions. Each package is named kernel, but each has a different qualifier.

For the x86 (32 bit) processors there three different kernel packages, one for Intel and AMD CPUs, one for the older Pentiums and AMD CPUs, and one for 32bit systems with more than 4 GIGS of memory. Each package will have a CPU reference in its filename: 686 for Intel (Pentium 2 and up, as well as Core Duos and up) and AMD (K8 and up), 586 for Pentium, K6, and other systems, and PAE for 32 bit systems with more than 4 GIGs.

```
kernel-2.6.version.i586.rpm
kernel-2.6.version.i686.rpm
kernel-PAE-2.6.version.i686.rpm
```

For 64-bit systems, like the Athlon 64 series and Intel Core 2 Duo, the 64-bit distribution will include only the x86_64 package.

```
kernel-2.6.version.x86_64.rpm
```

Note: The Xen virtualization, SMP (multi-processor support) and kdump (support for system dumps used for debugging) have been integrated into current kernels.

For each kernel, there are also corresponding kernel header packages (also known as builds), denoted with the term **devel**, that contain only the kernel headers. These are used for compiling kernel modules or software applications that do not need the full kernel source code, just the headers. The headers for your current kernel are already installed. The kernel headers will be installed in the **/etc/src/kernels** directory with a **build** link to it in the kernel's **/lib/modules** directory.

Installing Kernel Packages: /boot

New kernel versions with recent patches will be installed for you as updates. You can use PackageKit to install other kinds of kernels like PAE. You can also create and manually install your own customized kernel.

Update kernel packages

The PackageKit Update System will automatically download and install new kernels as they become available, selecting the version appropriate for your system. Normally you do not have to do anything to have the latest kernels installed, except to approve the install. They will be installed for you by the Package Updater. The previous kernel will remain installed, in case you have problems with the new kernel. In the event of problems, you can select the older kernel from the GRUB menu and use that instead.

The GRUB boot loader configuration file will be automatically edited for you to boot the new kernel. An entry for the new kernel will be placed in the **/boot/grub/grub.conf** file, keeping the entry for the previous kernel. All other entries and configurations will remain. Only the entry for the new kernel is added.

Install different kernel packages

If you want to install a particular kernel, not just an update of the currently installed version, you can use PackageKit or the **yum** command to select and install the new kernel. You may have to do this if you want to install a different kind of kernel, like the PAE kernel (large memory addressing). On PackageKit, search for "kernel" to list your kernel packages. With yum, just enter the pattern **kernel*** to list available packages. Install the one you want.

```
yum list kernel*
```

PackageKit and the **yum** command use YUM which will automatically select and install any dependent packages, as well as keep track of any needed updates. For this reason PackageKit and the yum command are the preferred way to install new kernels.

Install customized kernel packages

Alternatively, you can also install a particular kernel RPM package using the **rpm** command in a terminal window. This method may be needed for customized kernel packages that may not be on any YUM repository. To make sure a kernel RPM package was downloaded without any errors and to verify its authentication, you can use the **rpm** command with the **--checksig** option (to authenticate the package, you need the public key):

```
rpm --checksig *rpm
```

You can now install the kernel. As a safety precaution, you should preserve your old kernel in case the new one does not work out for some reason. This involves installing with the install (**-i**) option instead of the update (**-U**) option, creating a separate RAM disk for the new kernel, and then modifying **grub.conf** to have GRUB start up using the new kernel.

```
rpm -ivh kernel-2.6.version.i686.rpm
```

GRUB modifications for the installed kernel

The kernels are installed in the **/boot** directory. Performing an **ls -l** operation on this directory lists all the currently installed kernels. A file for your old kernel and a file for your new one now exist. When a new kernel is installed, the GRUB boot loader configuration file (**/boot/grub.conf**) is automatically modified to add the entry to invoke the new kernel. The kernel boots using the selected **/boot/vmlinuz-***version* kernel file. In your **grub.conf** file, there will be a line to reference this kernel file.

There will also be a line for the RAM disk, **initrd**. If your system has an RAID disk or any other specialized hardware, RPM will also create a RAM disk to hold appropriate support modules (you can create a RAM disk manually with the **mkinitrd** command). The RAM disk is named **initrd-***kernel-version***.img** and is located in the **/boot** directory, as in **/boot/initrd-2.6.***version***.img**.

```
kernel /boot/vmlinuz-version ro root=/dev/sda3
initrd /boot/initrd-version.img
```

If you are installing a customized kernel manually, you may have to make these modifications yourself.

Tip: Although it is not included with Fedora, user-mode Linux (UML) is an optional version of the kernel designed to run as a stand-alone program separate from the kernel.

In effect, it creates a virtual machine with disk storage implemented on a user file. UML is often used to test software or experiment with kernel configurations, without harming the real system. You can also use UML to implement virtual hosting, by running several virtual machines on one physical host. With a virtual machine, you can control the access to the host system, providing greater security. You can find out more about user-mode Linux at **user-mode-http://linux.sourceforge.net**.

Precautionary Steps for Modifying a Kernel of the Same Version

If you want to modify your kernel configuration and build a new one, you should be sure to give the kernel a different name. This way you current kernel will not be overwritten. As a precaution, you could make a backup copy of your current kernel. In case something goes wrong with your modified version, you can always boot from the copy you kept. You do not have to worry about this happening if you are installing a new version of the kernel. New kernels are given different names, so the older one is not overwritten.

Boot Loader manual configuration

Installation of a kernel package will automatically create a GRUB boot loader entry for the new kernel. You will be able to select it on startup. Entries for your older kernel will remain. Should you create a customized version of the current kernel, while keeping the original versions as a backup, you would then need to create a new entry for the original kernel in the boot loader configuration file. Leaving the entry for the original kernel is advisable in case something goes wrong with the new kernel. This way, you can always reboot and select the original kernel. For example, in **/boot/grub.conf**, add a new entry, similar to the one for the old kernel, which references the new kernel in its kernel statement. The **grub.conf** entry would look something like the following code. You could then select the entry with the title "Old Linux (2.6.*version*.orig)" at the GRUB menu to launch the original kernel.

```
title Original Linux (2.6.version.orig)
 root (hd0,2)
 kernel /boot/vmlinuz-2.6.version.orig root=/dev/sda3
 initrd /boot/initrd-2.6.version.orig.img
```

If you use a label for the boot partition, the **root** option for the **kernel** statement would look like this for a boot partition labeled **/**:

```
kernel /boot/vmlinuz-2.6.version.orig ro root=LABEL=/
```

Boot disk creation

You should also have a boot CD-ROM ready, just in case something goes wrong with the installation (**mkbootdisk** package). With a boot CD-ROM, you can start your system without using the boot loader. You can create a boot CD-ROM using the **mkbootdisk** utility. To create a boot CD-ROM, you need to know the full version number for your kernel. You can, in fact, have several kernels installed, and create boot CD-ROMs for each one (your **grub.conf** file lists your kernel version number). If the kernel version is 2.6.*version*, use it as the argument to the **mkbootdisk** command to create a boot CD-ROM for your system:

To make a boot CD-ROM, you can use the **--iso** option with the **--device** option to specify the CD image file. You can then burn the image file to a CD-ROM with an application like K3b. In the next example, the user creates a CD-ROM image file, called **myimage.iso**, for a boot CD-ROM of the 2.6.*version* kernel:

```
mkbootdisk --iso --device myimage.iso  2.6.version
```

Creating a Kernel from Source Code

Instead of installing already-compiled binary versions of the kernel, you can install the kernel source code on your system and use it to create the kernel binary files yourself. Kernel source code files are compiled with the **gcc** compiler just as any other source code files are. One advantage to compiling the kernel is that you can enhance its configuration, adding support for certain kinds of devices.

Be sure to consult the Fedora Project documentation on building a kernel at:

```
http://fedoraproject.org/wiki/Building_a_custom_kernel
```

Installing Kernel Sources with Fedora SRPM

You can obtain the kernel source code package for Fedora either through Yum or by downloading it directly from the Fedora distribution's **SRPMS** directory. The source code package is not included with the binary RPMS files.

To download directly from the Fedora distribution site, first access the Fedora distribution's FTP or mirror site, using your Web browser or FTP client. Then change to the **source** directory. The following entry is the complete URL for the kernel source package on the Fedora distribution site.

```
http://download.fedora.redhat.com/pub/fedora/linux/releases/10/Fedora/source/SRPMS/kernel-2.6.27.5-117.fc10.src.rpm
```

To install the package, you can use the **rpm** command in a terminal window as shown here. With a Web browser, you could also install using the software installer option on download.

```
rpm -ivh kernel-2.6.27.5-117.fc10.src.rpm
```

Alternatively, you can download the source package using Yum. You will need to enable the Fedora source code repository. Unless you want to download source code for applications on a regular basis, you may want to just enable the source code repository for the download of the kernel source code package. To do this you run Yum using the **yum** command in a terminal window, with the **--enablerepo** option assigned **fedora-source**.

```
--enablerepo=fedora-source
```

You will also need supporting **rpmbuild** tools in the **rpm-build** package, as well as additional rpm development tools in the **rpmdevtools** package. Also make sure the **yum-utils** package is installed. This package contains the **yumdownloader** tool which you use to download just a specific rpm package, not install it. You will use **yumdownloader** to download the source code rpm package.

```
rpm-build
rpmdevtools
yum-utils
```

You can now download the kernel source using **yumdownloader** and the **--source** option. The **--source** option instructs **yumdownloader** to access the Fedora source code repository. The package name is simply **kernel**. You can use the **--destdir** option to specify a download directory.

```
yumdownloader --enablerepo=kernel-source --source kernel
```

You then should make sure that you have all the tools and packages installed that you will need to build the kernel source. To do this, run the **yum-builddep** command on the kernel source RPM package. You will have to run this command as the root user. You can use the **su -c** command to run just that command as root.

```
su -c 'yum-builddep kernel-2.6.27.5-117.fc10.src.rpm'
```

Note: The Fedora source repository is configured in the **/etc/yum.repos.d/fedora.repo** file, in the **[fedora-source]** section. The **enabled** option is set to 0 by default, disabling the repository. To enable it, you can edit the **fedora.repo** file and set the **enabled** option to 1.

Building the Kernel Source

Fedora RPM sources are managed through the Fedora RPM build directories, not in the traditional **/usr/src** directory. It is always recommended that you **not** manage and build the kernel source as the **root** user. This means that you should not install the RPM source kernel package directly as the **root** user, which would place the files in **/usr/src/redhat** directory. This directory should be avoided for use for kernel sources.

Instead you should always install the kernel source in a local user directory as a local user, not the root user (the same is true for kernel archive files downloaded from **http://kernel.org**). You still need to use the **rpmbuild** set of directories like SPEC and BUILD. To set this local build environment up, you use the **rpmdev-setuptree** command (**rpmdevtools** package). Run this command once logged in as the user you want to use for managing the kernel source.

```
rpmdev-setuptree
```

This command will create an **rpmbuild** directory with subdirectories for SPEC, Build, RPMS, SOURCES, and SRPMS. The directory is built at **$(HOME)/rpmbuild**, with **$(HOME)** evaluating to your user home directory.

Move the **kernel** RPM package in the **rpmbuild** directory, and then run the **rpm** command to extract it, **-Uvh**. Error message will appear saying that it is using root, but it really is not. You can ignore the error message.

```
rpm -Uvh kernel-2.6.27.9-159.fc10.src.rpm
```

The RPM spec file, **kernel-2.6.spec**, will be placed in the **SPEC** directory. Configuration information, patches, and the compressed archive for the kernel source are placed in the **SOURCES** directory.

To generate the source code, you then use the **`rpmbuild`** command with the kernel spec file and specify the type of architecture you want. Change to the **SPECS** directory and run

rpmbuild with the **-bp** option (build) and the **--target** option to specify the architecture. The following command will extract the i686 version of the kernel:

```
rpmbuild -bp --target=i686  kernel.spec
```

The extracted kernel source will be placed in the **BUILD** directory under **kernel-2.6.27**, with subdirectories for the **i686** version and the **vanilla** version, **linux-2.6.27.i686** and **vanilla-2.6.27**. You can change to i686 directory and then run commands like **make xconfig** to configure the kernel source.

Once you have finished modifying the kernel, you then both generate the kernel binary and package it into an RPM package for installation, using the **rpmbuild** command.

Different Kernel System Configurations

Should you want to use a different configuration, you need to configure your kernel source for that particular variation. To do this, you use the appropriate configuration file for your systems. These are located in the kernel source's **configs** directory. For the i386 RPM source you will find kernel configuration files for the x86 systems, i586 and i686

The **.config** file will already be the version you specified in the target option when you extracted the source with **rpmbuild**. For a different configuration, you copy the config file you want as the **.config** file in the top kernel source directory. The following example copies the i686 configuration file as the kernel source configuration file:

```
cp configs/kernel.2.6.27-i686.config  .config
```

Then update your kernel source with the new configuration by issuing the following command:

```
make oldconfig
```

Tip: If you are using the original kernel source, you should also check for any patches.

Configuring the Kernel

Once the source is installed, you can configure the kernel. Configuration consists of determining the features for which you want to provide kernel-level support. These include drivers for different devices, such as sound cards and SCSI devices. You can configure features as directly included in the kernel itself or as modules the kernel can load as needed. You can also specifically exclude features. Features incorporated directly into the kernel make for a larger kernel program. Features set up as separate modules can also be easily updated. Documentation for many devices that provide sound, video, or network support can be found in the **/usr/share/doc** directory. Check the **kernel-doc** package to find a listing of the documentation provided.

```
rpm -ql kernel-doc
```

Note: If you configured your kernel previously and now want to start over from the default settings, you can use the **make mrproper** command to restore the default kernel configuration.

Kernel Configuration Tools

You can configure the kernel using one of several available configuration tools: **config**, **menuconfig**, **xconfig** (qconf), and **gconfig** (gkc). You can also edit the configuration file directly. These tools perform the same configuration tasks but use different interfaces. The **config** tool is a simple configure script providing line-based prompts for different configuration options. The **menuconfig** tool provides a cursor-based menu, which you can still run from the command line. Menu entries exist for different configuration categories, and you can pick and choose the ones you want. To mark a feature for inclusion in the kernel, move to it and press the SPACEBAR. An asterisk appears in the empty parentheses to the left of the entry. If you want to make it a module, press M and an *M* appears in the parentheses. The **xconfig** option runs qconf, the QT (KDE)–based GUI kernel configuration tool, and requires that the QT libraries (KDE) be installed first. The **gconfig** option runs the **gkc** tool, which uses a GTK interface, requiring that GNOME be installed first. Both **qconf** and **gkc** provide expandable menu trees, selectable panels, and help windows. Selectable features include check buttons you can click. All these tools save their settings to the **.config** file in the kernel source's directory. If you want to remove a configuration entirely, you can use the **mrproper** option to remove the **.config** file and any binary files, starting over from scratch.

```
make mrproper
```

You start a configuration tool by preceding it with the **make** command. Be sure you are in the kernel directory (either your local **rpmbuild/BUILD** directory you are using for an RPM kernel package, or the local directory you used for the compressed archive, such as **tar.gz**). The process of starting a configuration tool is a **make** operation that uses the Linux kernel Makefile. The **xconfig** and **gconfig** tools should be started from a terminal window on your window manager. The **menuconfig** and **config** tools are started on a shell command line. The following example lists commands to start **xconfig**, **gconfig**, **menuconfig**, and **config**:

```
make gconfig
make xconfig
make menuconfig
make config
```

gconfig (gkc)

The GTK kernel configuration tool (**gkc**) is invoked with the **gconfig** option. This uses a GNOME-based interface that is similar to **xconfig** (**qconf**). The **gkc** tool opens a Linux Kernel Configuration window with expandable submenus like those for qconf (see Figure 7-1). Many categories are organized into a few major headings, with many now included under the Device Drivers menu. The Load and Save buttons and File menu entries can be used to save the configuration or to copy it to a file. Single, Split, and Full view buttons let you display menus in one window, in a display panel with another panel to containing an expandable tree to select entries, or as a single expandable tree of entries. The Expand button will expand all heading and subheading, whereas Collapse will let you expand only those you want displayed. Use the down and side triangles for each entry to expand or collapse subentries.

Clicking an entry opens a window that lists different features you can include. Entries are arranged in columns listing the option, its actual name, its range (yes, module, or no), and its data (yes, no, or module status). Entries in the Options menu let you determine what columns to display:

Name for the actual module name; Range for the selectable yes, no, and module entries; and Data for the option status, titled as Value.

The Range entries are titled N, M, and Y and are used to select whether not to include an option (N), to load it as a module (M), or to compile it directly into the kernel (Y). Entries that you can select will display an underscore. Clicking the underscore will change its entry to Y for module or direct kernel inclusion, and N for no inclusion. The Value column will show which is currently selected.

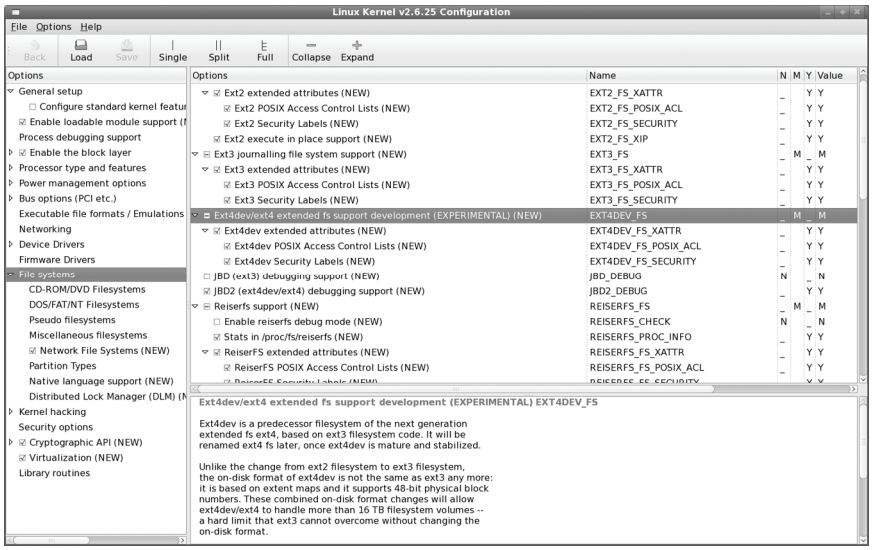


Figure 7-1: gconfig kernel configuration (make gconfig)

The Options column will include a status showing whether the option is included directly (check mark), included as a module (line mark), or not included at all (empty). To quickly select or deselect an entry, double-click the option name in the Options field. You will see its check box checked, lined (module), or empty. Corresponding N, M, and Y entries for no inclusion, module, or kernel inclusion are selected. The default preference for either module or direct kernel inclusion for that option is selected automatically. You can change it manually if you wish.

menuconfig

The **`menuconfig`** option generates a cursor based text menu. You can use arrow, tab, and enter keys to move to different menus and entries (see Figure 7-2). Use the spacebar to select or deselect and entry.

xconfig (qconf)

The **`xconfig`** option invokes the qconf tool, which is based on KDE QT libraries. KDE has to first be installed. The qconf tool opens a Linux Kernel Configuration window listing the different configuration categories. It has a slightly simpler interface, without the expand or collapse buttons or the columns for module and source status that **`gconfig`** has.

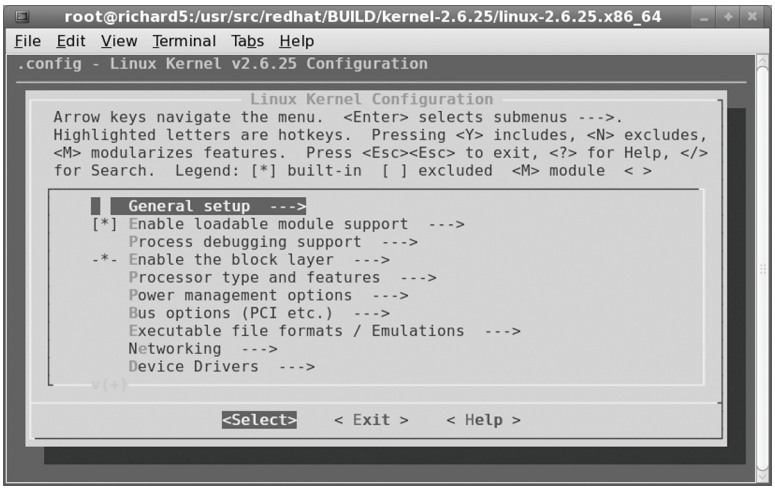


Figure 7-2: menuconfig kernel configuration (make menuconfig)

Important Kernel Configuration Features

The **`xconfig`**, **`menuconfig`**, and **`gconfig`** tools provide excellent context-sensitive help for each entry. To the right of each entry is a Help button. Click it to display a detailed explanation of what that feature does and why you would include it either directly or as a module, or even exclude it. When you are in doubt about a feature, always use the Help button to learn exactly what it does and why you would want to use it. Many of the key features are described here. The primary category for a feature is listed in parentheses.

Tip: As a rule, features in continual use, such as network and file system support, should be compiled directly into the kernel. Features that could easily change, such as sound cards, or features used less frequently should be compiled as modules. Otherwise, your kernel image file may become too large and slower to run.

- **Loadable Module Support** In most cases, you should make sure your kernel can load modules. Click the Loadable Module Support button to display a listing of several module management options. Make sure Enable Loadable Module Support is checked. This feature allows your kernel to load modules as they are needed. Kernel Module Loader should also be set to Yes, because this allows your daemons, such as your Web server, to load any modules they may need.
- **Processor Type And Features** Processor Type And Features entries allow you to set up support for your particular system. Here, you select the type of processor you have and processor features.
- **Power Management Features** Enable power management support like CPU scaling and ACPI.
- **Device Drivers** The Devices Drivers lists entries that enable support for your SATA, SCSI, video, sound, and USB devices among others

- **Multi-Device Support (RAID and LVM) (Device Drivers)** Multi-Device Support lists entries that enable the use of RAID devices. You can choose the level of RAID support you want. Here you can also enable Logical Volume Management support (LVM), which lets you combine partitions into logical volumes that can be managed dynamically.
- **SCSI Support (Device Drivers)** If you have any SCSI devices on your system, make sure the entries in the SCSI Support window are set to Yes. You enable support for SCSI disks, tape drives, and CD-ROMs here. The SCSI Low-Level Drivers window displays an extensive list of SCSI devices currently supported by Linux. Be sure the ones you have are selected.
- **Network Device Support (Device Drivers)** The Network Device Support window lists several general features for network device support. There are entries here for windows that list support for particular types of network devices, including Ethernet (10 or 100Mb) devices, token ring devices, WAN interfaces, and AppleTalk devices. Many of these devices are created as modules you can load as needed. You can elect to rebuild your kernel with support for any of these devices built directly into the kernel.
- **Multimedia Devices (Device Drivers)** Multimedia devices provide support for various multimedia cards like DVB as well as Video4Linux.
- **File Systems** The File Systems window lists the different types of file systems Linux can support. These include the Windows file system VFAT (Windows 95/98), but not NTFS (this managed by the ntfs3g module). DVD/CD-ROM file systems such as ISO and UDF are listed. UDF support is configured as a module that needs to be loaded. Network File Systems lists entries like NFS, CIFS (Samba), and NCP (NetWare) are included> Miscellaneous filesystems HFS (Macintosh).
- **Sound (Device Drivers)** For Sound you can select the Advanced Linux Sound Architecture sound support, expanding it to the drivers for particular sound devices (the Open Sound System is also included, though deprecated).
- **Kernel Hacking** The Kernel Hacking window lists features of interest to developers who work at the kernel level and need to modify the kernel code. You can have the kernel include debugging information, and also provide some measure of control during crashes.
- **Cryptographic API** Cryptographic listed supported encryption ciphers.
- **Virtualization** Virtualization lists support for KVM, processor builtin virtualization support.

Once you set your options, save your configuration. The select the Save entry on the File menu to overwrites your **.config** configuration file. The Save as option lets you save your configuration to a particular file.

Compiling and Installing an RPM Packaged Kernel

Kernels you are modifying using the Fedora RPM packages, with the rpmbuild directories, will be compiled and installed by creating an RPM kernel package that will contain the new kernel binaries. You then simply install that RPM package to install the new kernel.

Once you have finished modifying the kernel, you then both generate the kernel binary and package it into an RPM package for installation. Use the **rpmbuild** command with the **-bb** option. The kernel RPM package will be placed in the RPMS directory.

```
rpmbuild -bb --target=i686  kernel.spec
```

You can then install that kernel package using the **rpm** command with the **-Uvh** options.

Often, when compiling software using the kernel source, the software expects to find the kernel source in the **/etc/src/linux** directory. To accommodate this situation you can create a link name **/etc/src/linux** to your linux kernel source directory. This is the directory you expanded the linux source package.

```
ln -s /home/myuser/rpmbuild/BUILD/kernel-2.6.27/linux-2.6.27.i686/
/usr/src/linux
```

Creating the Kernel from the Original Source Code

You can also download the original kernel source form **www.kernel.org**. This version will not be optimized for Fedora. It should be placed in directory of your choosing, but not in the **/usr/src/linux** directory.

These versions are normally much more recent than those available on for Fedora, but they may not have been thoroughly tested on the Fedora platform. The kernel source is in the form of compressed archives (**.tar.gz**). They have the prefix **linux** with the version name as the suffix. You decompress and extract the archive with the following commands. You first change to the local directory you chose and then unpack the archive with either Fileroller or the **tar** command. It creates a directory with the prefix **linux** where the source files are placed. The following example extracts the 2.6.27.6 kernel:

```
cd mykernel
tar -xzvf linux-2.6.27.6.tar.gz
```

Be sure to unpack the archive for the **kernel.org** version in a directory you choose, like **mykernel** in a home directory. The source will reside within a subdirectory that has the prefix **linux** and a suffix consisting of the kernel version, as in **linux-2.6.27** for kernel 2.6, revision 27. The local directory in this example would be **mykernel/linux-2.6.27**.

You then can modify the kernel using the kernel configuration tools as described in the previous section, Kernel Configuration Tools. Use the **make** command with the **gconfig**, **xconfig**, or **menuconfig** options to start a configuration interface.

```
make gconfig
```

Compiling and Installing a Kernel from the Original Source Code

To compile and install a kernel using the original source code, you run a series of **make** operations, first to compile the kernel and then to install it and its modules. Now that the configuration is ready, you can compile your kernel. You can display the options available with the help option.

```
make help
```

You also have to clean up any object and dependency files that may remain from a previous compilation. Use the following command to remove such files:

```
make clean
```

For a more complete removal, including the config file, use **mrproper**.

```
make mrproper
```

Option	Description
install	Creates the kernel and installs it on your system.
bzImage	Creates the compressed kernel file and calls it **bzImage**.
isoimage	Creates the kernel and installs it on a CD-ROM disk (creates a boot disk).
vmlinuz	Creates the bare kernel image file.
fdimage	Creates floppy disk image with the kernel, 1.44 MB (bootable).
fdimage288	Creates floppy disk 2.88 image with the kernel (bootable).

Table 7-1: Compiling Options for Kernel make command

You can use several options to compile the kernel (see Table 7-1). The **bzImage** option simply generates a kernel file called **bzImage** and places it in the **arch** directory. For Intel and AMD systems, you find **bzImage** in the **86/boot** subdirectory, **arch/86/boot**. For a kernel source, this would be in **arch/86/boot**. A kernel type directory will also be generated in the **arch** directory like **i686** or **x86_64**, which will hold the **boot** directory only with a link named **bzImage** to the **arch/86/boot/bzImage** file.

```
make bzImage
```

The options in Table 7-1 create the kernel, but not the modules—those features of the kernel to be compiled into separate modules. To compile your modules, use the **make** command with the **modules** argument.

```
make modules
```

To install your modules, use the **make** command with the **modules_install** option. This installs the modules in the **/lib/modules/***version-num* directory, where *version-num* is the version number of the kernel. You should make a backup copy of the old modules before you install the new ones.

```
make modules_install
```

The **install** option both generates the kernel files and installs them on your system as **vmlinuz**, incorporating the **make bzImage** step. This operation will place the kernel files such as **bzImage** in the **/boot** directory, giving them the appropriate names and kernel version number.

```
make install
```

The commands for a simple compilation and installation are shown here:

```
make clean
make
```

```
make modules_install
make install
```

If you want, you could enter these all on fewer lines, separating the commands with semicolons, as shown here:

```
make clean; make; make modules_install; make install
```

A safer way to perform these operations on single lines is to make them conditionally dependent on one another, using the **&&** command. In the preceding method, if one operation has an error, the next one will still be executed. By making the operations conditional, the next operation is run only if the preceding one is successful.

```
make clean && make && make modules_install &&  make install
```

Installing a Kernel Image File Manually

To install a kernel **bzImage** file manually, copy the **bzImage** file to the directory where the kernel resides and give it the name used on your distribution, such as **vmlinuz-2.6.*version***. Remember to first back up the old kernel file, as noted in the precautionary steps. **vmlinuz** is a symbolic link to an actual kernel file that will have the term **vmlinuz** with the version name. So, to manually install a **bzImage** file, you copy it to the **/boot** directory with the attached version number such as **vmlinuz-2.6.*version***.

```
make bzImage
cp arch/i386/boot/bzImage /boot/vmlinuz-2.6.version
```

You will also have to make a copy of the **System.map** file, linking it to the **System.map** symbolic link.

```
cp arch/i386/boot/System.map  /boot/System.map-2.6.version
```

The following commands show a basic compilation and a manual installation. First, all previous binary files are removed with the **clean** option. Then the kernel is created using the **bzImage** option. This creates a kernel program called **bzImage** located in the **arch/i386/boot** directory. This kernel file is copied to the **/boot** directory and given the name **vmlinuz-2.6.*version***. Then a symbolic link called **/boot/vmlinuz** is created to the kernel **vmlinuz-2.6.*version*** file. Finally, the modules are created and installed:

```
make clean
make
make modules_install
cp arch/i386/boot/bzImage /boot/vmlinuz-2.6.version
cp System.map /boot/System.map-2.6.version
```

You will also have to create a module ram disk image and add an entry for the new kernel in the GRUB bootloader configuration file, **/etc/grub/grub.conf**. On Fedora, you can perform these operations automatically with the **new-kernel-pkg** command. Use the **--mkinitrd** to create a module RAM disk, and **--install** to set up a GRUB entry. The **--depmod** option runs **depmod** to generate a current dependency list for kernel modules.

```
new-kernel-pkg -v --mkinitrd  --install --depmod  vmlinuz-2.6.version
```

Module RAM Disks

If your system uses certain block devices unsupported by the kernel, such as some SCSI, RAID, or IDE devices, you will need to load certain required modules when you boot. Such block device modules are kept on a RAM disk that is accessed when your system first starts up (RAM disks are also used for diskless systems). For example, if you have a SCSI hard drive or CD-ROMs, the SCSI drivers for them are often held in modules that are loaded whenever you start up your system. These modules are stored in a RAM disk from which the startup process reads. If you create a new kernel that needs to load modules to start up, you must create a new RAM disk for those modules. You need to create a new RAM disk only if your kernel has to load modules at startup. If, for example, you use a SCSI hard drive but you incorporated SCSI hard drive and CD-ROM support (including support for the specific model) directly into your kernel, you don't need to set up a RAM disk (support for most IDE hard drives and CD-ROMs is already incorporated directly into the kernel).

Tip: If you are experimenting with your kernel configurations, it may be safer to put a new kernel version on a boot CD-ROM, rather than installing it on your system. If something goes wrong, you can always boot up normally with your original kernel still on your system (though you can always configure your boot loader to access previous versions).

If you need to create a RAM disk, you can use the **mkinitrd** command to create a RAM disk image file. The **mkinitrd** command incorporates all the IDE, SCSI, and RAID modules that your system uses, including those listed in your **/etc/modules.conf** file (the **new-kenrel-pkg** script will also run **mkinitrd**). See the Man pages for **mkinitrd** and RAM disk documentation for more details. **mkinitrd** takes as its arguments the name of the RAM disk image file and the kernel that the modules are taken from. In the following example, a RAM disk image called **initrd-2.**6.**version.img** is created in the **/boot** directory, using modules from the 2.6.27 kernel. The 2.6.27 kernel must already be installed on your system and its modules created.

```
mkinitrd /boot/initrd-2.6.version.img 2.6.27.5-117.fc10.x86_64
```

You can select certain modules to be loaded before or after any SCSI module. The **--preload** option loads before the SCSI modules, and **--with** loads after. For example, to load RAID5 support before the SCSI modules, use **--preload=raid5**:

```
mkinitrd --preload=raid5 raid-ramdisk 2.6.27.5-117.fc10.x86_64
```

In the **grub.conf** segment for the new kernel, place an **initrd** entry specifying the new RAM disk:

```
initrd /boot/initrd-2.6.27.5-117.fc10.x86_64.img
```

GRUB Boot Loader Configurations

For the boot loader GRUB, you can configure your system to enable you to start any of your installed kernels. As seen in the section "Precautionary Steps for Modifying a Kernel of the Same Version," you can create an added entry in the boot loader configuration file for your old kernel. As you install new kernel versions, you could simply add more entries, enabling you to use any of the previous kernels. Whenever you boot, your boot loader will then present you with a list of kernels to choose from. For example, you could install a developmental version of the kernel, along with a current stable version, while keeping your old version. In the image line for each

entry, you specify the filename of the kernel. You can create another boot loader entry for your older kernel (the **new-kernel-pkg** script will also add an entry in the **grub.conf** file for the new kernel).

In the next example, the **/boot/grub.conf** file contains entries for two Linux kernels, one for the kernel installed earlier, **2.6.27.5-117.fc10.x86_64**, and one for a more recent kernel, **2.6.27.9-159.fc10.x86_64** With GRUB, you only have to add an new entry for the new kernel.

```
# grub.conf generated by anaconda
#
#boot=/dev/sda
default=0
timeout=30
splashimage=(hd0,2)/boot/grub/splash.xpm.gz
title New Fedora (2.6.27.9-159.fc10.x86_64)
  root (hd0,2)
  kernel /boot/vmlinuz-2.6.27.9-159.fc10.x86_64 ro root=/dev/sda3
  initrd /boot/initrd-2.6.27.9-159.fc10.x86_64.img
title  Old Fedora (2.6.27.5-117.fc10.x86_64)
  root (hd0,2)
  kernel /boot/vmlinuz-2.6.27.5-117.fc10.x86_64 ro root=/dev/sda3
  initrd /boot/initrd-2.6.27.5-117.fc10.x86_64.img
title Windows XP
   rootnoverify (hd0,0)
   chainloader +1
```

Kernel Boot Disks

Instead of installing the kernel on your system, you can simply place it on a boot disk or CD-ROM and boot your system from that disc. For a DVD/CD-ROM you can use the **isoimage** option. For a floppy disk image you can create either a 1.44 or 2.88 image (which will hold the 2.6 kernel). Use the **fdimage** option for a 1.44 image and **fdimage288** for the 1.88 image. Both fdimage and fdimage288 create corresponding floppy disk images in the in the **arch/i386/boot** directory. The **fdimage288** image is often used for virtual users.

```
make isoimage
make fdimage
make fdimage288
```

8. Virtualization

Virtual Machine Manager: virt-manager

Kernel-based Virtualization Machine (KVM): Hardware Virtualization

Xen Virtualization

There are now several methods of virtualization available for use on Fedora. These range from the para-virtualization implementation employed by Xen to the hardware acceleration used by the Kernel-base Virtualization Machine (KVM) for Intel and AMD processors with hardware virtualization support. All of these can be installed and managed easily with the Virtual Machine Manager, a GNOME based tool that provides a simple GUI interface for managing your virtual machines and installing new ones. Fedora also provides the GNOME VM applet, **gnome-applet-vm**, a panel applet that can monitor your virtual machines. See Table 8-1 for a listing of virtualization resources. All virtualization packages are available from the Fedora repository.

Resource	Description
http://fedoraproject.org/wiki/Virtualization_Quick_Start	Virtualization on Fedora, both KVM and Xen with details on Virtual Machine Manager, Xen controller is not supported in Fedora 10
http://fedoraproject.org/wiki/Tools/Virtualization	Fedora virtualization resources
http://virt-manager.et.redhat.com/	Virtual Machine Manger, virt-manager.
http://bellard.org/qemu/	QEMU software virtualization
http://kvm.qumranet.com/kvmwiki	KVM hardware virtualization
http://libvirt.org/	libvirt tool kit for accessing Linux virtualization capabilities.
http://sourceforge.net/projects/kvm	Kernel Virtual Machine (KVM) Web site
http://kraxel.fedorapeople.org/xenner/	Xenner Xen emulation using KVM
http://fedoraproject.org/wiki/Features/VirtStorage	Virtual storage on Fedora

Table 8-1: Virtualization Resources

All virtualization methods can be installed and managed with the Virtual Machine Manager. The Virtual Machine Manager greatly simplifies the process of installing and managing virtual operating systems (guest OS). With just a few steps you can install Windows or other Linux distributions on your Fedora system, and run them as guest operating systems whenever you need them. KVM virtual hosts will run directly from the processor, running almost as fast and as stable as if you installed it separately with a dual boot configuration.

There are two major methods currently used for virtualization, full and para virtualization. Full virtualization (KVM or QEMU) runs a guest OS independently. Since Fedora 9, this is the only kind of virtualization supported. Even Xen, which no longer uses a separate kernel, can only be run as a KVM host. Hardware virtual support in the CPU processors is required for Xen. Later developments may restore software para-virtualization support. See **http://virt.kernelnewbies.org** for general virtualization links and overview.

Virtual Machine Manager: virt-manager

You can easily manage and set up KVM or Xen virtual machines using the Virtual Machine Manger (virt-manager). Be sure that virt-manger is installed. Select Virtual Machine Manager from the Applications | System Tools menu. This will display a window listing your

virtual machines (see Figure 8-1). Features like the machine ID, name, status, CPU and memory usage will be displayed. You can use the View menu to determine what features to display. Click the Help entry in the Help menu to show a detailed manual for Virtual Machine Manager.

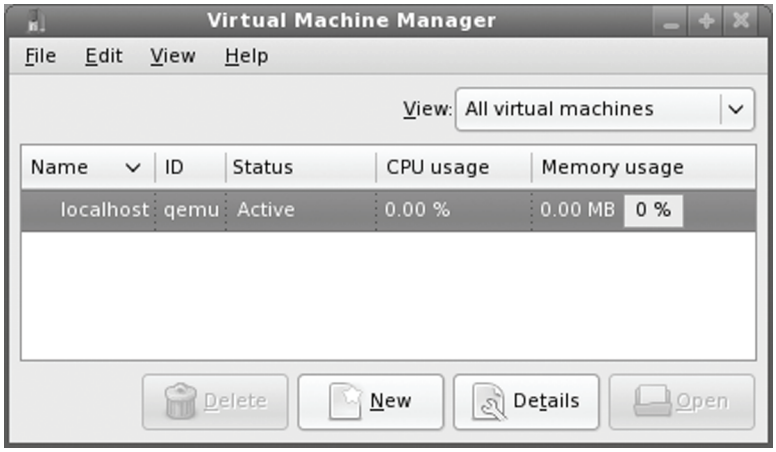


Figure 8-1: The Virtual Machine Manager

For detailed information about the host machine, click Host Details from the Edit menu. The Overview panel will show information like the host name, the number of CPUs it has, and the kind of hypervisor it can launch. The Virtual Network panel shows your virtual networks, listing IPv4 connection information, the device name, and the network name. A default virtual network will already be set up. Select guest and click Details button for guest information.

You can have several hosts. This is the case if you have installed both Xen and KVM support. KVM will show a qemu host and Xen will show a Xen host. Initially the hosts are disconnected. Your first have to connect to the host to access or create its virtual machines. To access a host, right-click on its entry in the Virtual Machine Manger window, and select connect from the pop-up menu. Once connected, the several of the buttons at the bottom of the widow become active. You can use the New button to create a new virtual machine.

To create a virtual machine, select New Machine from the File menu. This will start up virt-install wizard. You will be prompted for name, the kind of virtualization, the location of the Operating System install disk or files, the storage to use for the guest operating system, and the amount of system memory to allocate for the guest OS.

After entering a name, you then choose your virtualization method. If you are running a standard kernel, you will only have the option to use a fully virtualized method. On systems with Intel VT and AMD SVM processors you will also have the option to enable hardware acceleration. This means using KVM (kernel-based virtual machine) support that will provide processor-level hardware virtualization. For processors without hardware virtualization support, software emulation is used.

Tip: With a system with extensive memory and processor support, you can even run or install guest operating systems simultaneously using KVM from the Virtual Machine Manager.

You then choose the location of the OS install media. For a fully virtualized OS, this can either be a disk image or a CD/DVD-ROM, like a Windows install disk. You then choose the type of operating system you are installing, first selecting a category like Linux or Windows, and then a particular distribution or version like Centos Linux or Windows XP.

You then choose the storage method. This can be either an existing partition or a file. If you choose a file, you can either set a fixed size (like a fixed partition), or have the file expand as needed. Should the file be on partition with a great deal of free space, this may not be an issue. Initially the file will be 4 GB, though you may want to make it larger to allow for regular use.

You then choose a virtual network or a physical device for your network connection. Then choose the amount of system memory to allocate to each virtual machine, as well as the number of virtual CPUs to use. A final screen displays all your configuration information for the new virtual machine before you start installation. You can still cancel at this point.

The Virtual Machine Console is then opened and the install process for that operating system begins (see Figure 8-2). You install as you normally would. To use your mouse on for installing the operating system, pass the mouse over the Virtual Machine Console window. It will be captured and you can use it on the operating system install dialogs. To return the mouse to your Fedora desktop, press Ctrl-Alt and move the mouse away from the Virtual Machine Console.

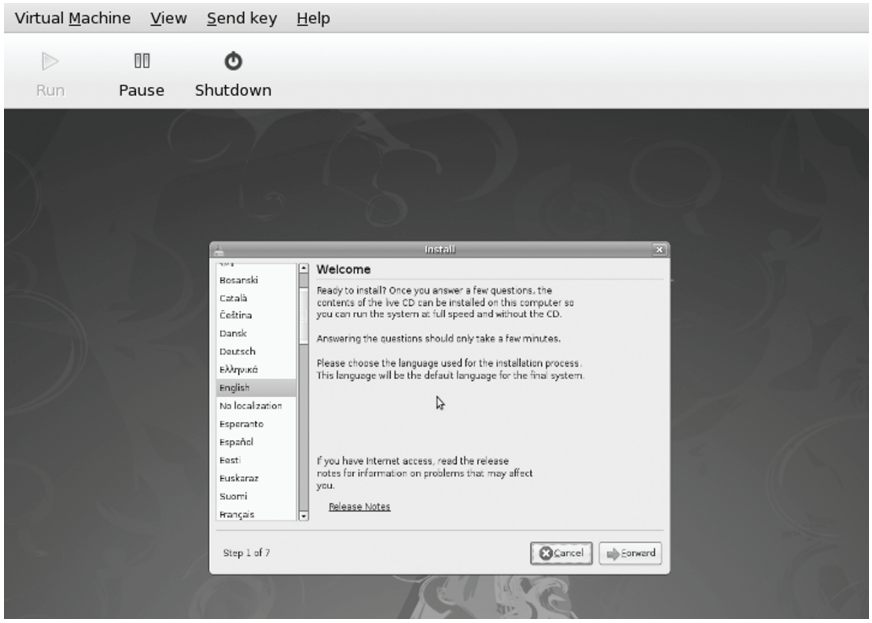


Figure 8-2: The Virtual Machine Console (installation)

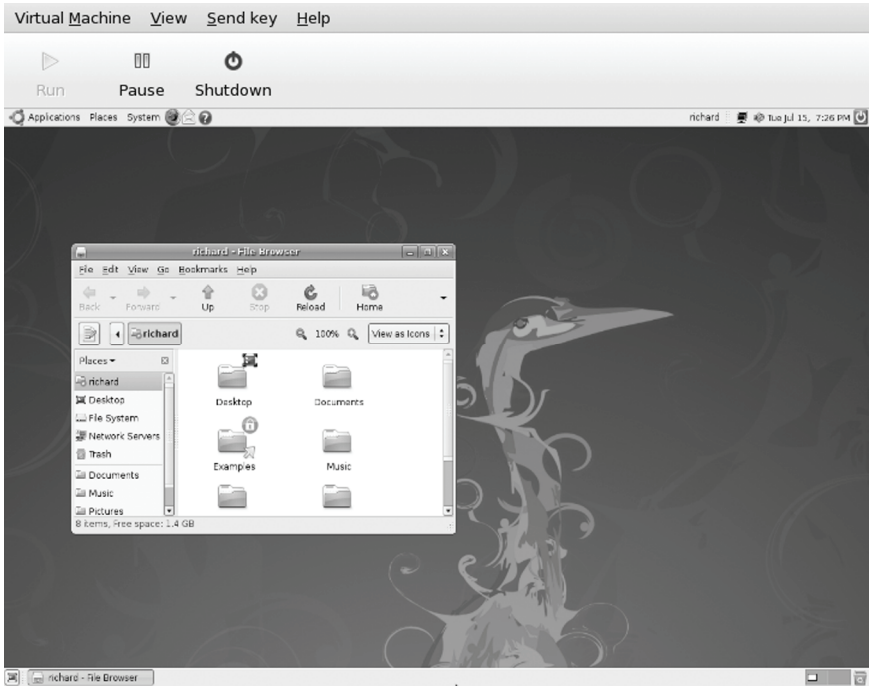


Figure 8-3: The Virtual Machine Console (running os)

Once you have finished your installation, that operating system will try to reboot. You man have to click the Run button on the Virtual Machine Console window to have it restart. There are buttons to run, pause, and shutdown the new operating system.

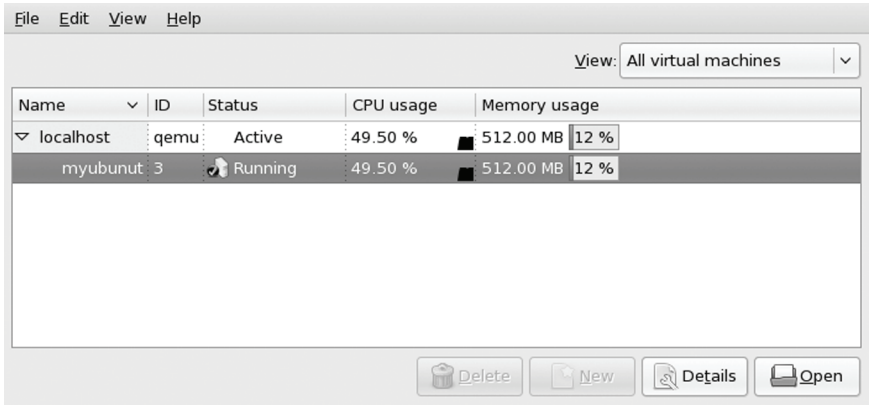


Figure 8-4: The Virtual Machine Console with Virtual Machines

Once restarted the operating system will start as it normally would, with its own login interface. Once logged in, you will have a fully functional operating system, like the one shown in Figure 8-2 for Ubuntu.

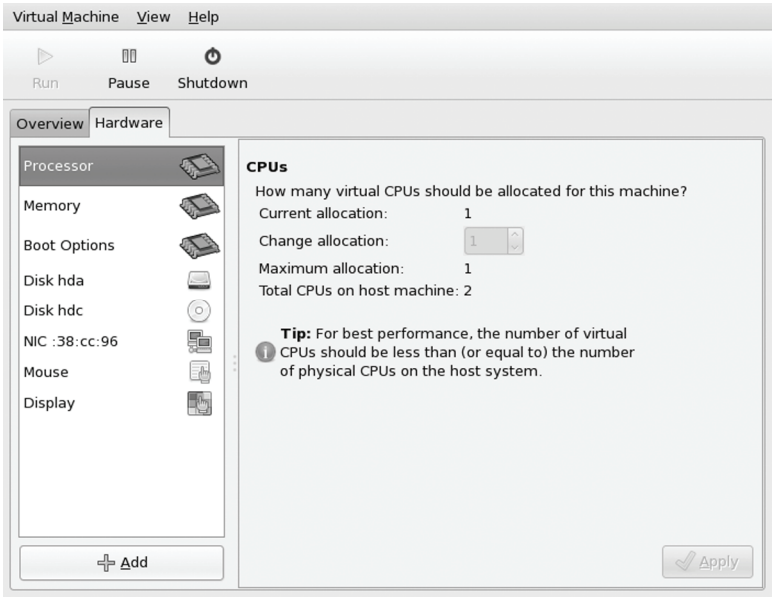


Figure 8-5: The Virtual Machine Details

To shut down, you should use the operating system's shut down commands, not the Virtual Console Machine's Shutdown button.

Your installed virtual machines will now be listed under its host in the Virtual Machine Manager window (See Figure 8-4). You can sue the Open button to start it up in the Virtual Machine Console.

To see more information about particular virtual machine, select its entry and click the Details button (see Figure 8-5). This opens a dialog with panes for Overview and Hardware. The Overview pane shows virtual machine name, its disk name, status, and a graph of its real time CPU usage. The Hardware pane shows information about the virtual devices as well as CPU and memory allocations.

Tip: You can also manage your virtual machines from the command line with **virsh**.

Storage Management

Virt-manager can manage storage through libvirt library. This allows the creation of remote virtual hosts using storage management to setup and access storage for them. Essentially libvirt is performing para-virtualized tasks in that it is accessing real storage devices from virtual machines.

Virt-manager organizes storage into pools and volumes. From the Edit menu, choose Connection Details. This opens the Host Details window. Click on the Storage Pools tab. Existing storage pools are listed to the left and configuration for the pools is displayed on the right pane. Configuration includes information like the storage location, current state, and volumes it holds. When you created a virtual machine using a file image, a default pool was set up with the volume

for that disk image. Its location would be **/var/lib/libvirt/images**, the default location for disk file images.

To create a new pool, first click on the + button on the lower left. This opens the "Add a New Storage Pool" window where you first enter the name and the type. You can have different types of pools. The type entry is a drop down menu listing available types. These include like a Filesystem Directory for a directory (dir), Physical Disk Device (disk) for hard disk partitions, Pre-Formatted Block Devices (fs) for partitions already formatted with a file system, iSCSI target (iscsi) for and ISCSI device, LVM Volume Group (logical) for an LVM file system, and Network Exported Directory (ntfs) for an NFS shared directory.

The next window has entries for target path for file system directories, Source Path for hard disk partitions and LVM volume groups, and Host Name for NFS servers. Entries will be grayed out depending on which is needed for a given storage pool type. Once created the new storage pool will appear on the Storage Pools tab.

Kernel-based Virtualization Machine (KVM): Hardware Virtualization

With kernel version 2.6.21, hardware virtualization is now directly supported in the kernel (pervious versions used a kernel module). Hardware virtualization is implemented by Intel and AMD as Hardware Virtual Machine abstraction layer. Intel processors that have hardware virtualization support are labeled VT (Virtualization Technology), and AMD processors are labeled SVM (Secure Virtual Machine). An HVM system has the capability to provide full virtualization, not requiring a specially modified version of an OS kernel like Xen's para-virtualization method uses. You can even run Windows XP directly from Linux using the HVM capability. KVM is an open source project developed by Qumranet, **http://kvm.qumranet.com/kvmwiki**. Be sure to check the KVM entries in Fedora Project virtual quick start guide for details on how to implement KVM, **http://fedoraproject.org/wiki/Docs/Fedora10VirtQuickStart**.

Kernel-based Virtual Machine (KVM) uses the hardware virtualization in a processor to run virtual machine directly from hardware. There is no underlying software translation. KVM operates directly with the processor.

Hardware requirements are as follows:

- An Intel (VT) or AMD (SVM) virtualization enabled processor (like AMD AM2 socket processors or Intel Core2Duo processors). You may need to enable virtualization support in your motherboard. Some motherboards will work better than others. In some cases you may have to disable ACPI support in the motherboard BIOS to allow Windows XP to run.
- At least 1 GIG of system memory to allow space for the virtual OS to run. The hardware virtual OS requires its own memory.

KVM is launched as a process directly from the Linux kernel, as if booting to a new OS. As a process is can be managed like any Linux process. KVM adds a quest process mode with its own user and kernel mode. This is in addition to the Linux kernel and user modes. KVM uses the kernel modules kvm-intel or kvm-amd to interface with the processor's virtualization hardware. A modified version of a software emulator QEMU is used to run the OS guest. QEMU was originally designed as an emulator and is also available as such for processors without hardware virtualization. See **http://fabrice.bellard.free.fr/qemu** form more information on QEMU.

Note: KVM is run with a modified version of QEMU which has limited virtual device support, like the graphics driver.

Start the Virtual Machine Manager on your Gnome desktop (Applications | System Tools | Virtual Machine Manager). Choose New Machine for the File menu. This starts the virt-install wizard. When choosing the type of virtualization to use, select Fully Virtualized and make sure hardware acceleration is selected (Enabled kernel/hardware acceleration). You are then prompted for various features like the name, the amount of system memory to use, whether to use a given partition or an image file along with the file size, graphics support, and where the install image is located (this can be a CD/DVD-ROM disk, though a disk image is preferred for Windows)).

Once installed, you can use the Virtual Machine Manager to start up your guest OS at any time. Your guest OK is run in a virtual machine console.

Note: To access data directly on your virtual disks or files, you can use **lomount** or **kpartx**.

Xen Virtualization Kernel

Fedora now includes versions of the kernel that incorporate Xen Virtualization. Xen Virtualization technology allows you to run different operating systems on a Linux system, as well as running virtual versions of the kernel to test new applications. Xen is an open source project run by the University of Cambridge Computer Laboratory in coordination with the Open Source Development Labs and several Linux distributors, including Fedora. You can find more about Xen at **www.cl.cam.ac.uk/Research/SRG/netos/xen**. Fedora integrates Xen support into all its kernels (there are no separate Xen kernel packages).

Xen is designed so that on a single Xen server you can run several virtual machines to run different operating systems at the same time. Xen is a para-virtualized system, meaning that the guest operation system has to be modified to run on Xen. It cannot run without modification as it could on a fully virtualized system like VMware. Xen is designed to use a para-virtualization approach to increase efficiency, giving its virtual machines nearly the same level of efficiency as the native kernel. For an operating system to work on Xen, it must be configured to access the Xen interface. Currently only UNIX and Linux operating systems are configured to be Xen compatible, though work is progressing on Windows.

To use the Xen kernel, you first have to install the Xen kernel package as well as the Xen server, tools, and documentation. There are one Xen kernel packages, which incorporates support for running Xen in domain 0 (xen0), as a server, and for unprivileged (xenU), user access.

The Xen support is now integrated into all kernel packages. In addition you can download install Xen tools and documentation packages. Detailed documentation will be in **/usr/share/xen-2**. Configuration files will be placed in **/etc/xen** directory, and corresponding kernels in the **/boot** and **/lib/modules** directories. In the **/etc/xen** directory you will find the **xend-config** file for configuring the Xen **xend** server, as well as example Xen configuration files.

```
xen
xen-hypervisor
xen-runtime
xen-watch
```

Xen sets up separate virtual machines called domains. When the Xen kernel starts up, it is designed to set up a primary domain, domain0, which manages your system and sets up virtual machines for other operating systems. The virtual machines it managers are domain U domains which provide user unprivileged access. Management of the virtual machines is handled by the **xend** server. Your native kernel would be installed on domain0, which will handle most of the hardware devices for all the other virtual machines. Xen is designed to use two kinds of domains, domain 0 (xen0) as a server and controller of other virtual systems, and domain U for unprivileged (xenU), user access. In Fedora 10, domain 0, the controller, is not supported. Only domain U, the unprivileged user operation, is supported. Xen can run as a virtual guest machine, but not as a controller or server.

You control the domains with the **xend** server. **xend** messages are placed in the **/var/log/xend.log** file. The xend server should automatically be started when you start up with the Xen kernel. There is a **xend** service script on Fedora letting you start the xend server manually with the following command. You can also manage xend from **system-config-services**:

```
/sbin/service xend start
```

Xenner

You can run Xen as a domain U, guest system, by using Xenner to emulate a Xen domain 0 controller. Xenner actually runs KVM, which requires hardware virtualization support in the CPU. Xenner uses KVM to emulate a Xen domain 0 server. It runs KVM but with Xen support. Xenner is still in development and may not be that reliable. You first enable and run Xenner as a service, which makes sure that KVM is loaded, and then rung several xenner emulation daemons for KVM.

```
service xenner start
```

You then use the xenner application to run a Xen virtual machine.

Should domain 0 support be integrated into the kernel by Fedora 11, you would no longer need Xenner. Fedora 8 still supports domain 0 for Xen, so if you are using Xen as a server, you may want to stay with Fedora 8 until Fedora 11.

Using Configuration Files

To create a virtual machine, you have to first create a configuration file for it in the **/etc/xen** directory. Here you will find sample configuration files you can use as a template. Here are settings you may want to change:

- **kernel** The path to the kernel image used by the virtual machine
- **root** The root device for the domain
- **memory** The amount of memory you will allow the domain to use
- **disk** The block devices (partitions) you want the domain to use
- **dhcp** Have the domain use DHCP to set networking. For manual configuration, you can set netmask and gateway parameters
- **hostname** The host name for the virtual machine

- **vif** The MAC address to use (random ones will be generated if none is specified)
- **extra** Additional boot parameters
- **restart** Automatic restart options: always, never, onreboot

You will have to set the kernel. This is the location of the xenU kernel. The root entry specifies the partition where the boot image is stored. The disk entry lets you specify disks that the new domain will use. These can be logical volumes or disk image files. These cannot be already mounted by the primary domain. Note that **xenguest-install** described later will set up a disk image file to be used by the virtual machine, instead of a logical volume. The following uses a disk image **/home/mynewvm1** which will appear to the virtual machine as **xvda** and have read/write access.

```
disk = [ 'phy:/home/mynewvm1,xvda,w' ]
```

You may also want to enable **dhcp** if your network uses that to set up a network connection. For example the kernel setting would look like this:

```
kernel = "/boot/vmlinuz-2.6.17-1.2145_FC6
```

You use the xm command with the **-create** option, followed by the configuration file. In this example there is a configuration file in **/etc/xen** called **my-unewvm1** which is the same name as the virtual machine. Check the man page for **xmdomain.cfg** for detailed configuration options and examples.

```
xm create my-newvm1
```

You can then connect to it with the **console** option.

```
xm console my-newvm1
```

You can combine the two xm using the **-c** option for the console connection.

```
xm create -c my-newvm1
```

You will still have to install the operating system you want to use. You can do this on a designated partition or logical volume, or use xenguest-install. You can install with a text based system or use VNC for a virtual install.

Hardware Virtual Machine: hvm

Xen also provides support for the Hardware Virtual Machine, hvm. This is the Hardware Virtual Machine abstraction layer that Intel is implementing in its new processors as Intel VT-x. AMD will implement hvm as SVM. An hvm system has the capability to provide full virtualization, not requiring a specially modified Xen compatible version of an OS kernel. Theoretically you could even run Windows XP on Xen using the hvm capability (the Windows XP installer throws up other complications unrelated to virtualization). Intel's virtualization for the Itanium processors is called VT-i. VT-x is for x86 processors.

Xen can support a system with an hvm capable processor. Example configuration file for hvm in the /etc/xen directory has the extension **.hvm**. In this file, options are set to detect and use hvm. The xenguest-install also checks for hvm.

virt-install

Instead of configuring a file directly, you can use the **virt-install** script. This script can be used to install Fedora or other Linux Xen compatible operating systems. It currently can only install from a remote network location using an **http://**, **nfs://**, or **ftp://** prefix. This script will now allow you to use less than 256MB for each virtual machine. If you want to use less memory, say for a scaled down version of Fedora, you will have to us the configuration files directly as described in the previous section. For Fedora, you can use the online Fedora repository.

If you have a limited amount of RAM memory, you may need to limit the amount the **domain 0** virtual machine is using. You can reduce this to the recommended 256MB with the following command.

```
xm mem-set 0 256
```

To start **virt-install** script, open a terminal window and enter the script name.

```
virt-install
```

You will be prompted to set different parameters, and then a configuration file will be automatically generated. You will first be prompted to name the virtual machine. This is your host name. Then you are asked how much RAM to allocate. A minimum of 256 is required. This means that for just one virtual machine you would have to have at least 500MB of RAM, 256MB for the Xen0 server (Domain 0) and 256MB for the guest/user machine, (Domain 1). More virtual machines would correspondingly more RAM.

You are then prompted for the disk path for the virtual machine image file. Enter the path with the image file name. You are then prompted for the size of the image file in Gigabytes. Virtual machines use an image file where its entire system is kept. Finally you are prompted for the location of installation files for the operating system you want to install. Here you can enter an FTP, Web, or NFS site, such as the online Fedora repository. Keep in mind that the script looks for the Xen compatible kernel images in the **images/Xen** directory of the distribution. Your download location has to prefix this directory. The online Fedora 10 directory is shown in the next example.

```
# virt-install
What is the name of your virtual machine? my-newvm1
 How much RAM should be allocated (in megabytes)? 256
 What would you like to use as the disk (path)? /home/my-newvm1
  How large would you like the disk to be (in gigabytes)? 8
Would you like to enable graphics support?(yes or no) yes
What is the install location?
http://download.fedora.redhat.com/pub/fedora/linux/releases/10/i386/os/
```

Once the files have been downloaded, the text-based install interface will start up, asking for keyboard and language. If you enabled graphics support, the standard graphical install will start up.

Managing Virtual Machines: xm

After you have installed the system, you can then create a connection to it with **xm** command. To access a particular domain, use the **console** option and the domain name.

```
xm console my-newvm1
```

If the domain no longer exists, you have to also create it and then connect.

```
xm create -c my-newvm1
```

Check the **xm** options on the **xm** man page for other operations you can perform on your virtual machines.

To access domains, you use the **xm** command. The **list** option lists your domains. The listing will include detailed information such as its domain ID, the CPU time used, memory used, and the domain state. The following lists your domains:

```
xm list
```

The **xm save** and **restore** options can be used to suspend and restart a domain.

Block devices such as partitions and CD-ROMs can be exported from the main domain to virtual domains. This allows a given virtual domain to use a particular partition. You can even share block devices between domains, though such shared devices should be read only.

To start and stop your domains, you can use the **xendomains** service script. The **xendomains** script will use xm with the **create** option to create a domain configured in the **/etc/xen/auto** directory. Place Xen domain configuration files in this directory for **xendomains** to start. The following command manually starts your domains:

```
service xendomains start
```

To have your domains start automatically at boot, use **chkconfig** with the **xendomains** script to add the script.

```
chkconfig --add xendomains
```

You can also manage your domains using the **xensv** web interface using port 8080. Start the **xensv** server with the following command.

```
xensv start
```

Note: Fedora also provides the Gnome VM monitor, **gnome-applet-vm**, a panel applet that can monitor your virtual machines. They make use of a libvirt library which provides an API for GUI access to Xen.

Part 2: Security

9. Authorization

Controlled Access with Policykit: Authorizations

Pluggable Authentication Modules

FreeIPA

Authorization, encryption, and permissions are all methods for controlling access. Authorizations can control access to administrative tools, making sure only valid and trusted users make changes to your system setup. Encryption can protect messages and files you may send, and digital signatures can confirm the source of a message or file. Users can also place their own access controls on their files using permissions and access control lists (ACLs). You can even encrypt entire file systems, making them accessible only with a valid key.

Certain security packages control access to resources such as devices, messages, directories, and file systems. PolicyKit provides controls for accessing devices and administrative tools by users. It is designed to permit limited administrative access to particular users, instead of allowing full root user access.

You can use encryption, integrity checks, and digital signatures to protect data transmitted over a network. For example, the GNU Privacy Guard (GPG) encryption as supported by Seahorse encryption management lets you encrypt your e-mail messages or files you want to send, as well as letting you sign them with an encrypted digital signature authenticating that the message was sent by you. The digital signature also includes encrypted modification digest information that provides an integrity check, allowing the recipient to verify that the message received is the original and not one that has been changed or substituted.

Permissions can be set on file and directories allowing access to just the owner, members of a group, or to all other users. This is the traditional method of controlling access to files. You can also use access control lists to add further restrictions. Access control lists provide more refined access, but are more difficult to manage. You can also encrypt entire file systems, using the same Public Key encryption method used for messages and archives.

Controlled Access with Policykit: Authorizations

Designed by the Freedesktop.org project, PolicyKit allow ordinary users and applications access to administrative controlled applications and devices. Currently it supports several key administrative operations including the Hardware Abstraction Layer (HAL) enabled devices, NetworkManager, PulseAudio, PackageKit, Samba, and the GNOME the clock and system monitor. Though this could be done with other operations like group permissions, PolicyKit aims to provide a simple and centralized interface for granting users access to administration controlled devices and tools. Currently PolicyKit is used to grant access to shared devices managed by HAL This includes most of the devices on your system including removable ones. It is not yet used to control access to administrative tools, like the system-config tools. For these you still PAM modules. Three important exceptions are PackageKit, NetworkManager, and PulseAudio. PackageKit manages software, Network Manger controls your network and wireless configurations, and Pulse Audio configures your sound devices.

PolicyKit can allow for more refined access. Instead of an all or nothing approach, where a user had to gain full root level control over the entire system just to access a specific administration tool, PolicyKit can allow access just to a specific administrative applications (currently only the GNOME clock is supported). All other access can be denied. Without PolicyKit, this kind of access could be configured for in a limited way for some devices, like mount and unmounting CD/DVD discs, but not for applications. A similar kind of refined control is provided with PAM and **sudo**, allowing access to specific administrative applications, but administrative password access is still required, and root level access, though limited to that application, is still granted. You can find out more about PolicyKit at **http://hal.freedesktop.org/docs/Policykit**.

A listing of applications and devices controlled by PolicyKit are shown here.

world-clock-applet	allow setting the time and date
Network Manager	Network detection
Power Management	shut down and reboot
PackageKit	Software Management
GNOME System Monitor	managing processes
PulseAudio	Sound device interface
storage devices	mount and unmount fixed and removable devices
device access	video, sound, scanners, PDA, DVB, CD/DVD-RW/ROM, firewire

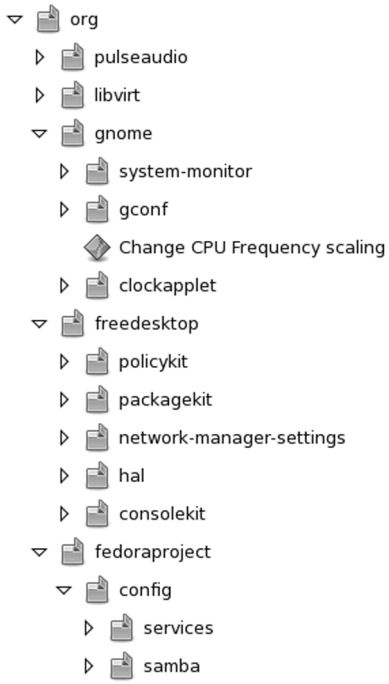


Figure 9-1: PolicyKit sidebar, collapsed

With PolicyKit, administration controlled devices and applications are set up to communicate with ordinary users, allowing them to request certain actions. If the user is allowed to perform the action, then the request is authorized and the action is performed. In the case of devices, which are now controlled and managed by HAL, request can be sent to HAL which then can authorize the action. Technically, all devices and administrative tools are considered mechanism to which requests are sent by user PolicyKit agents. Administration mechanisms use a

shared library called **policykit** to decide whether to grant access. Users and application requests are validated by a **libpolkit-grant** shared library. The **policykit** will check the validations provided by **libpolkit-grant** and allow access accordingly.

Authentication can be required just for the user (user password) or for an administrator (root user password). Access can further be controlled by time limits: indefinite, for the rest of the current session, or as long as the process is active.

PolicyKit Agent

To gain access, a user makes use of an authentication agent. The PolicyKit GNOME agent is installed with PolicyKit and can be run by any administration enabled user from System | Preferences | System | Authorization. This runs the **polkit-gnome-authorization** tool with which you can set policykit access. Both the **policykit** and **gnome-policykit** packages are the Fedora repository and installed by default. You can also run the PolicyKit agent from the terminal window by entering the following command.

```
polkit-gnome-authorization
```

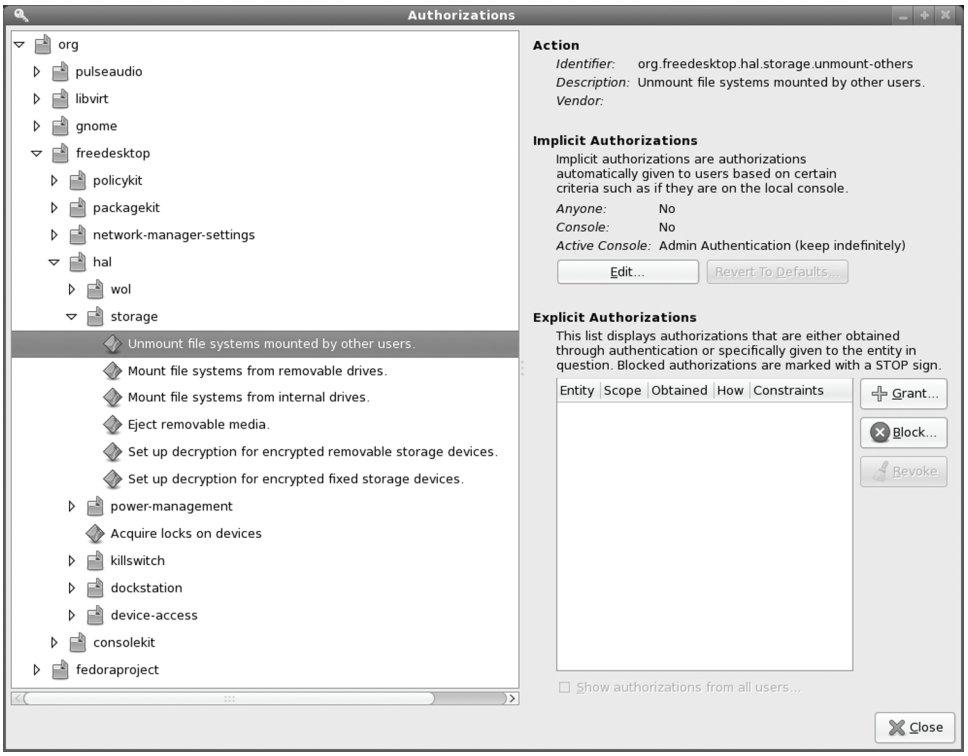


Figure 9-2: storage devices in hal

The PolicyKit GNOME agent will be displayed. When you try to make any changes, an Authenticate dialog will open, prompting you to enter the **root** user password. This will give **root** user authorization to make the change. You can choose to remember authorization. Click the Authenticate button to make the change.

PolicyKit sidebar

The PolicyKit agent will display a window with a sidebar showing and expandable tree of supported PolicyKit devices and applications. Collapsing the tree gives you a better view of what is available (see Figure 9-1). There are main entries for gnome and freedesktop. The freedesktop entry holds freedesktop supported tools including hal, policykit, PackageKit, and network-manager.

Application entries let you control access to certain tasks. The PolicyKit entry lets you control access for installing, removing, and updating software, as well as changing repository configuration. The clock applet entry controls changes to the system time and time zone. The System Monitor entry has controls for processes (ending and priority changes). The Network Manager entry controls access to the modification of the network connections.

The hal entries let you control access to power management, storage, and devices (see Figure 9-2). The storage section lets you control mounting for internal and removable drives, as well as encryption configuration.

Figure 9-3: PolicyKit Implicit and Explicit Authorizations

PolicyKit Implicit and Explicit Authorizations

A selected entry will display its PolicyKit configuration on the right panel. This is divided into three segments, Action, Implicit Authorizations, and Explicit Authorizations (see Figure 9-3).

Implicit Authorizations are applied to the device or tool for all users. These are set for Anyone, Console, and Active Console. For single user systems like most laptops, the logged in user will be the active console. Console and Anyone would cover remote users. The Yes entry allows

complete automatic access, and the NO entry denies all access. You can use the Edit button to change the settings (see Figure 9-4). You have the choice to restrict access to administrative users or users that provide their password, as well as limiting authorization to the duration of the session.

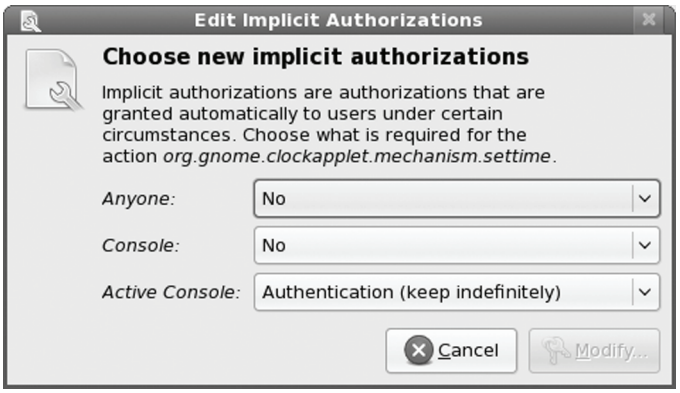


Figure 9-4: Implicit Authorization

Menu of possible authorizations for the Implicit authorizations is shown in Figure 9-5.

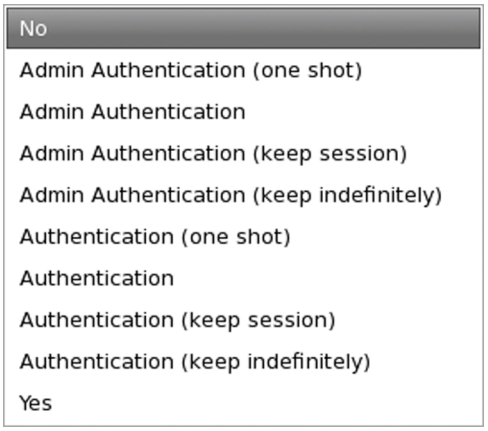


Figure 9-5: Implicit Authorization options

To grant access to specific users, click on the Grant button on the Explicit Authorizations panel. This opens a window where you can select a user and then specify the level of access (see Figure 9-6).

In the device-access section you can permit access by remote users to certain devices like video, sound, and DVB devices. In this case the Implicit Authorizations could be modified to allow access by Anyone. You could, instead, allow access to specific users.

In the freedesktop section, you can use entries in the policykit section to configure PolicyKit. Here you can specify who can revoke, read, modify, or grant PolicyKit authorizations.

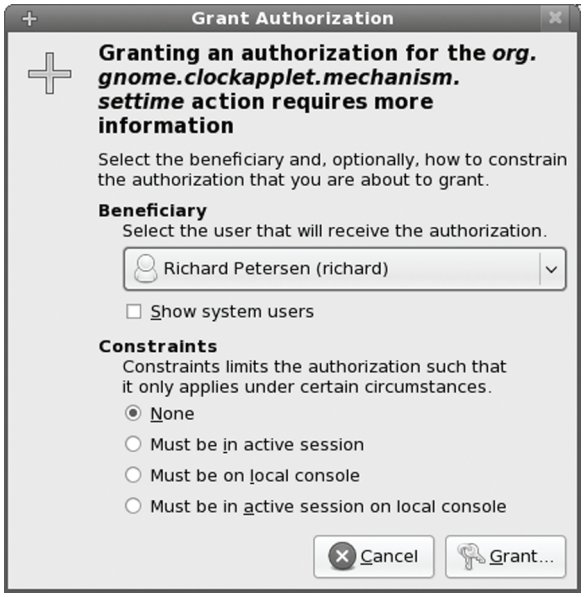


Figure 9-6: Explicit Authorization, Grant access

PolicyKit configuration files and tools

For devices and administrative applications to make use of PolicyKit, they have to be modified to access it. Currently HAL, which controls access to most devices, provides PolicyKit control for devices. On GNOME, the clock applet is configured for PolicyKit control. PolicyKit for devices and applications are configured using XML files with the extension **.policy** in the **/usr/share/PolicyKit/policy** directory. Here you will find **.policy** files for the gnome-clock-applet as well as several for HAL and one for PolicyKit.

The **/etc/PolicyKit/PolicyKit.conf** file is used to permit overriding any PolicyKit authorizations for users. Currently the configuration file is set up to always allow access to the root user and to any users with administrative access (admin group). It can be configured for specific users.

Though you would use polkit-gnome-authorization to configure PolicyKit, several command line tools are available. These include **polkit-auth** to mange authorization, **polkit-action** to list and modify allowed actions, **polkit-policy-file-validate** to validate a PolicyKit policy file, and **polkit-config-file-validate** to validate the PolicyKit configuration file. Should you make changes directly to the **PolicyKit.conf** file, you should run **polkit-config-file-validate** to make sure the file is valid. If you add or modify any of the .policy files, you can run **polkit-policy-file-validate** on them to verify that they are correctly configured.

system-config-authentication

To confirm that user identities are valid, your network may provide several authentication services. These can be enabled on your system using system-config-authentication (the name of the system-config-authentication program is **authconfig-gtk**). You can invoke system-config-authentication by selecting Authentication from the System | Administration menu. The system-config-authentication tool consists of three tabs, User Information, Authentication, and Options (see Figure 9-7). Configuration consists primarily of specifying the address of the service's server on your network. The User Information tab is used for services like NIS and LDAP, which maintain configuration information about systems and users on your network. The Authentication tab lists services for authenticating users. If your network maintains LDAP, Kerberos, and Samba PDC authentication servers, you can enable support for them here, specifying their servers and domains. The Options tab let you set authentication options like the Shadow and MD5 Passwords for password files. These are normally selected by default and provide password protection. From a pop-up menu you can select the password encryption codec. SHA512 is the default. Other options provide for more controlled access, like not creating user home directories until the user first logins, or checking **/etc/security/access.conf** for users to deny or allow access.

Figure 9-7: system-config-authentication (System | Administration | Authorization)

Pluggable Authentication Modules

Pluggable Authentication Modules (PAM) is an authentication service that lets a system determine the method of authentication to be performed for users. In a Linux system, authentication has traditionally been performed by looking up passwords. When a user logs in, the login process

looks up their password in the password file. With PAM, users' requests for authentication are directed to PAM, which in turn uses a specified method to authenticate the user. This could be a simple password lookup or a request to an LDAP server, but it is PAM that provides authentication, not a direct password lookup by the user or application. In this respect, authentication becomes centralized and controlled by a specific service, PAM. The actual authentication procedures can be dynamically configured by the system administrator. Authentication is carried out by modules that can vary according to the kind of authentication needed. An administrator can add or replace modules by simply changing the PAM configuration files. See the PAM Web site at **www.kernel.org/pub/linux/libs/pam** for more information and a listing of PAM modules. PAM modules are located in the **/lib/security** directory.

PAM Configuration Files

PAM uses different configuration files for different services that request authentication. Such configuration files are kept in the **/etc/pam.d** directory. For example, you have a configuration file for logging into your system (**/etc/pam.d/login**), one for the graphical login (**/etc/pam.d/gdm**), and one for accessing your Samba server (**/etc/pam.d/samba**). A default PAM configuration file, called **/etc/pam.d/other**, is invoked if no services file is present. The **system-auth** file contains standard authentication modules for system services generated by system-config-authentication and is invoked in many of the other configuration files. In addition, Fedora sets up an authentication for its configuration tools, such as system-config-services and system-config-network.

PAM Modules

A PAM configuration file contains a list of modules to be used for authentication. They have the following format:

```
module-type control-flag module-path module-args
```

The *module-path* is the module to be run, and *module-args* are the parameters you want passed to that module. Though there are a few generic arguments, most modules have their own. The *module-type* refers to different groups of authentication management: account, authentication, session, and password. The account management performs account verification, checking such account aspects as whether the user has access, or whether the password has expired. Authentication (**auth**) verifies who the user is, usually through a password confirmation. Password management performs authentication updates such as password changes. Session management refers to tasks performed before a service is accessed and before it is shut down. These include tasks like initiating a log of a user's activity or mounting and unmounting home directories.

Tip: As an alternative to the **/etc/pam.d** directory, you could create one configuration file called the **/etc/pam.conf** file. Entries in this file have a service field, which refers to the application that the module is used for. If the **/etc/pam.d** directory exists, **/etc/pam.conf** is automatically ignored.

The *control-flag* field indicates how PAM is to respond if the module fails. The control can be a simple directive or a more complicated response that can specify return codes like **open_err** with actions to take. The simple directives are **requisite**, **required**, **sufficient**, and **optional**. The **requisite** directive ends the authentication process immediately if the module fails to authenticate. The **required** directive only ends the authentication after the

remaining modules are run. The **sufficient** directive indicates that success of this module is enough to provide authentication unless a previous required module has failed. The **optional** directive indicates the module's success is not needed unless it is the only authentication module for its service. If you specify return codes, you can refine the conditions for authentication failure or success. Return codes can be given values such as **die** or **ok**. The **open_err** return code could be given the action **die**, which would stop all authentication operations and return failure.

An **include** directive simply includes the entries from the specified PAM module. Certain basic PAM configuration files like **system-auth** (system administrator authentication) are included in most of the other PAM configuration files. The system-auth file is generated automatically and includes key authentication modules like **pam_unix.so** and **pam_deny.so** for account access and password use. The **/etc/pam.d/vsftpd** configuration file for the FTP server is shown here. There are **include** entries for the **system-auth** configuration file.

```
#%PAM-1.0
session optional  pam_keyinit.so force revoke
auth    required  pam_listfile.so item=user sense=deny
                 file=/etc/vsftpd.ftpusers onerr=succeed
auth    required  pam_shells.so
auth    include   system-auth
account include   system-auth
session include   system-auth
session required  pam_loginuid.so
```

FreeIPA

FreeIPA provides authentication service for users on a local network. It centralizes identity, policy, and audit tasks (IPA) on a FreeIPA server. Users are no longer authenticated on separated hosts, but as part of a larger local network. FreeIPA pulls together several services to implement the user authentication service. This includes the Fedora Directory server (FDS), Kerberos authentication service, and DNS discovery service. The FDS provides an LDAP server with a Web interface and secure connections; LDAP provides user information management for users on your network. Kerberos provides authenticated security. DNS discovery makes access across the network automatic. Detailed documentation for FreeIPA is available at **www.freeipa.org**. Fedora specific information is located at **http://fedoraproject.org/wiki/Features/freeIPA**.

On the Fedora repository are the FreeIPA client, server, and admin-tools packages. Each host will need the FreeIPA client package installed, with only the server needing the FreeIPA server and admin-tools. The packages begin with the prefix **ipa**, as in **ipa-client**, **ipa-admintools**, and **ipa-server**.

fedora

10. Encryption

Public-Key Encryption
Managing keys with Seahorse
GNU Privacy Guard: gpg
Using GnuPG
Encrypting and decrypting data with gpg command
Signing Messages

Encrypting data is the only sure way to secure data transmitted over a network. Encrypt data with a key, and the receiver or receivers can later decrypt it. To fully protect data transmitted over a network, you should not only encrypt it, but also check that it has not been modified, as well as confirm that it was actually created by the claimed author. An encrypted message could still be intercepted and modified and then re-encrypted. Integrity checks such as modification digests make sure that the data was not altered. Though encryption and integrity checks protect the data, they do not authenticate it. You also need to know that the person who claimed to send a message actually is the one who sent it, rather than an imposter. To authenticate a message, the author can sign it using a digital signature. This signature can also be encrypted, allowing the receiver to validate it. Digital signatures ensure that the message you receive is authentic.

This type of encryption was originally implemented with Pretty Good Privacy (PGP). Originally a privately controlled methodology, it was handed over to the Internet Engineering Task Force (IETF) to support an open standard for PGP called OpenPGP (see Table 10-1). Any project can use OpenPGP to create encryption applications, such as GnuPG. Commercial products for PGP are still developed by the PGP Corporation, which also uses the OpenPGP standard.

Website	Description
www.gnupg.org	GnuPG, Gnu Privacy Guard
www.openpgp.org	IETF open standard for Pretty Good Privacy (PGP)
www.pgp.com	PGP Corporation, Pretty Good Privacy commercial products

Table 10-1: PGP Sites

Public-Key Encryption

Encryption uses a key to encrypt data in such a way that a corresponding key can decrypt it. In the past, older forms of encryption used the same key to both encrypt and decrypt a message. This, however, involved providing the receiver with the key, opening up the possibility that anyone who obtained the key could decrypt the data. Public-key encryption uses two keys to encrypt and decrypt a message, a private key and a public key. The *private* key you always keep and use to decrypt messages you have received. The *public* key you make available to those you send messages to. They then use your public key to encrypt any message they want to send to you. The private key decrypts messages, and the public key encrypts them. Each user has a set of private and public keys, securely kept in keyrings. Reciprocally, if you want to send messages to another user, you first obtain the user's public key and use it to encrypt the message you want to send to the user. The user then decrypts the messages with their private key. In other words, your public key is used by others to encrypt the messages you receive, and you use other users' public keys to encrypt messages you send to them. All the users on your Linux system can have their own public and private keys. They will use the **gpg** program to generate them and keep their private key in their own keyrings.

Digital Signatures

A *digital signature* is used to both authenticate a message and provide an integrity check. Authentication guarantees that the message has not been modified—that it is the original message sent by you—and the integrity check verifies that it has not been changed. Though usually

combined with encrypted messages to provide a greater level of security, digital signatures can also be used for messages that can be sent in the clear. For example, you would want to know if a public notice of upgrades of a Fedora release was actually sent by Fedora and not by someone trying to spread confusion. Such a message still needs to be authenticated and checked to see if it was actually sent by the sender or, if sent by the original sender, was not somehow changed en route. Verification like this protects against modification or substitution of the message by someone pretending to be the sender.

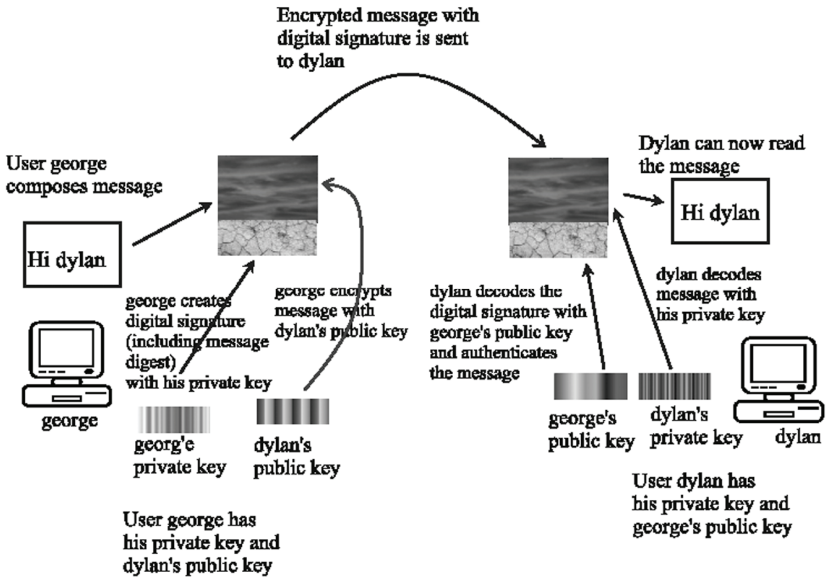


Figure 10-1: Public-key encryption and digital signatures

Integrity Checks

Digitally signing a message involves generating a checksum value from the contents of the message using an encryption hash algorithm such as the SHA2 modification digest algorithm. This is a unique value that accurately represents the size and contents of your message. Any changes to the message of any kind will generate a different value. Such a value provides a way to check the integrity of the data. The value is commonly known as the MD5 value, reflective of the MD5 hash algorithm that was used encrypt the value. The MD5 algorithm has since been replaced by the more secure SHA2 algorithms.

The MD5 value is then itself encrypted with your private key. When the user receives your message, they decrypt your digital signature with your public key. The user then generates an MD5 value of the message received and compares it with the MD5 value you sent. If they are the same, the message is authenticated—it is the original message sent by you, not a false one sent by a user pretending to be you. The user can use GnuPG (described in the next section) to decrypt and check digital signatures.

Combining Encryption and Signatures

Normally, digital signatures are combined with encryption to provide a more secure level of transmission. The message is encrypted with the recipient's public key, and the digital signature is encrypted with your private key. The user decrypts both the message (with their private key) and then the signature (with your public key). The user then compares the signature with one that user generates from the message to authenticate it. When GnuPG decodes a message, it will also decode and check a digital signature automatically. Figure 10-1 shows the process for encrypting and digitally signing a message.

Managing keys with Seahorse

For GPG and SSH encryption, signing, and decryption of files and text, GNOME provides Seahorse. With Seahorse you can manage your encryption keys stored in keyrings as well as SSH keys and passphrases. You can import keys, sign keys, search for remote keys, and create your own keyrings, as well as specify keyserver to search and publish to. All these operations can also be performed using the **gpg** command described later in this chapter. Seahorse is not installed by default. Install the **seahorse** and **seahorse-gnome** packages.

Though on the Fedora repository, Seahorse is not installed as part of the standard Fedora 10 installation. You will have to install is using PackageKit (System | Administration | Add/Remove Software).

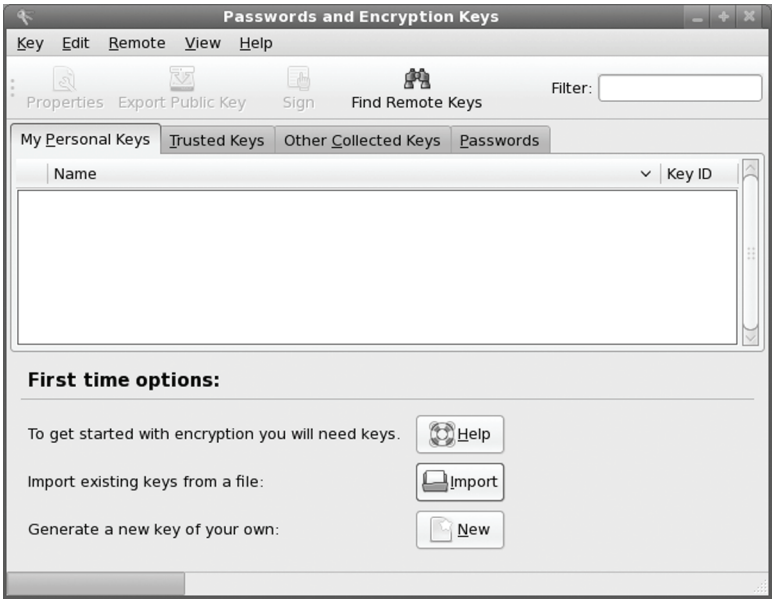


Figure 10-2: Seahorse First Time Use

Passwords and Encryption Keys

To import, sign, and locate keys you use the Password and Encryption Keys utility (Applications | Accessories | Passwords and Encryption Keys). The Passwords and Encryption

Keys window displays four tabs: My Personal Keys, Trusted Keys, Other Collected Keys, and Passwords. When you first start up the utility it will display three buttons on the lower part of the tab: Help, Import, and New (see Figure 10-2).

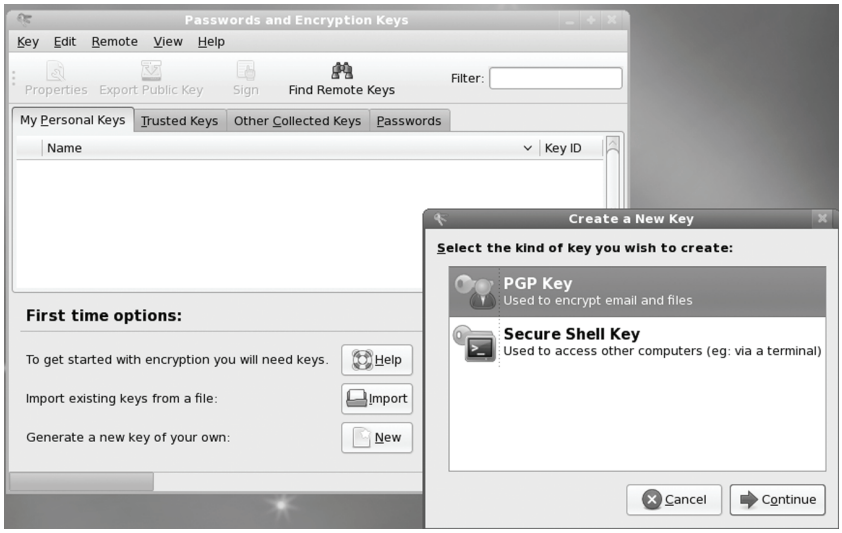


Figure 10-3: Choose Encryption key type

Creating a new private key

Click the New button (or choose Key | Create New Key), to create your own private/public keys. Keep in mind that before you can perform any encryption, you first have to set up your own GPG key pair, private and public. You first choose whether to set up a PGP or SSH key. The PGP entry sets up a GPG key (GPG is the GNU version of PGP). Choose the PGP Key entry and click Continue (see Figure 10-3).

Figure 10-4: Create Encryption key

This opens a New PGP Key window where you enter your name and email address. Click the Advanced key options drop-down arrow to set Encryption type (DSA, or RSA), Key strength, and Expiration Date (see Figure 10-4). Then click the Create button.

You are then asked to enter a passphrase (password) for the encryption key (see Figure 10-5). This passphrase will allow you to decrypt any data encrypted by your key.

Figure 10-5: Passphrase for encryption key

The key is then generated. This can take some time. During the generation process, a busy notification will let you know the generation is still in process (see Figure 10-6).

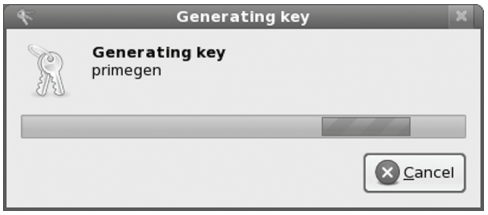


Figure 10-6: Generating encryption key

Once you key is created, it will appear in the My Personal Keys tab of the Passwords And Encryption Keys window (See Figure 10-7).

Importing Public Keys

In the Passwords And Encryption Keys window, click Use the Import button (or choose Import from the Key menu) to import any public keys you may downloaded. If you know the name of the key file, you can try searching the key servers for it. Choose Find Remote Keys from the Remote menu to open the Find Remote Keys dialog where you can enter a search string for the key (see Figure 10-8). The search term is treated as a prefix, matching on all possible completions. An expandable tree lists you key servers, letting you choose which ones to search. Results are then listed in a new window. Select the one you want, and then either click Import to import the key directly, or click Save Key As to save the key as an asc key file that you can later import. To see information about a key, select it and click the Properties button. Information about the owner and the key is displayed.

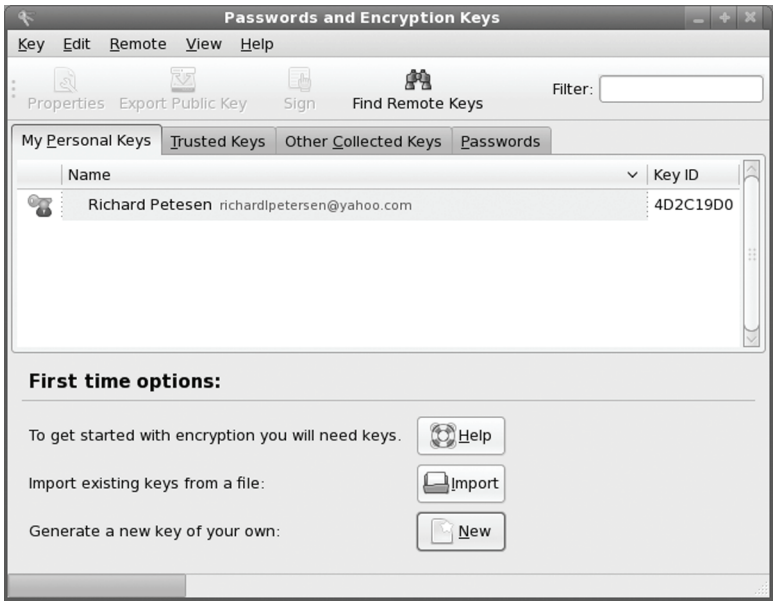


Figure 10-7: My Personal Keys

Figure 10-8: Searching for keys

Once you have imported the key, it will appear in the Other Collected Keys tab in the Passwords And Encryption Keys window (see Figure 10-9). If you know that you can trust the key, you can sign it, making it a trusted key. Right-click on its entry and choose Sign to open a signing

dialog, or click the Sign button. You are asked to specify how carefully you have checked this key (Not at all, Casually, and Very Carefully). The key will be moved from the Other Collected Keys tab to the Trusted Keys tab. It will then appear in the Trusted Keys tab.

Figure 10-9: Imported keys

When you created your own private key, you also generated a corresponding public key that other can use to decrypt data encrypted with that key. To make you public key available to others, you can export it to a file to send directly other users, automatically share it with other uses on your system, or publish on a keyserver. To export your public key to file, select your key in the "My Personal Keys" tab, and click the Export Public Key button. You can do this for your public keys also.

Seahorse integrates support for GPG. Should you import a key with the **gpg** command, it will appear in the Other Collected Keys tab. Also, you can sing a key using the **gpg** command with the **--sign-key** option and the key will appear in the Trusted Key tab.

Seahorse Settings

To manage and configure key support, you use the seahorse-preferences utility (System | Preferences | Encryption and Keyrings). The Password and Encryption Settings window opens with two tabs: Encryption and PGP Passphrases. On the Encryption tab you can select the default keyring. On the Passphrases tab you can set remembrance options, like setting a time period in which to remember the passphrases, or to remember them whenever you are logging in.

GNU Privacy Guard: gpg

To protect messages and text files, Fedora, like most Linux distributions, provides GNU Privacy Guard (GnuPG) encryption and authentication (www.gnupg.org). GnuPG is the GNU open source software that works much like Pretty Good Privacy (PGP) encryption. It is the

OpenPGP encryption and signing tool (OpenPGP is the open source version of PGP). With GnuPG, you can both encrypt and digitally sign your messages and files—protecting the message or file and authenticating that it is from you.

To protect messages that you send by e-mail, Evolution and KMail both support GnuPG encryption and authentication, along with Thunderbird with added GPG plugins. On Evolution, you can select PGP encryption and signatures from the Security menu to use GnuPG (the PGP options use GnuPG). On KMail, you can select the encryption to use on the Security panel in the Options window. For Thunderbird you can use the **enigmail** extension to support OpenGPG and PGP encryption (**http://enigmail.mozdev.org**). Be sure to always send a copy of your encrypted message to yourself. This way you can decrypt the message and read them as sent messages.

You can encrypt text files using GEdit as well as your Nautilus file manager. Other applications like Openoffice.org will support digital signatures, authenticating your files.

GNU Privacy Guard (GnuPG) operations are carried out with the **gpg** command, which uses both commands and options to perform tasks. Commonly used commands and options are listed in Table 10-2. Some commands and options have short forms that use only one hyphen. Normally, two hyphens are used. If you just want to verify the validity of a digital signature, you can use **gpgv** instead. This is a stripped-down version of **gpg** used just for signature verification.

For managing your encryption keys, you can use the GNOME Seahorse encryption management tool (Applications | Accessories | Passwords and Encryption Keys), instead of **gpg** commands directly (see the previous section). Encryption for text files and GEdit text editor are provided by the Seahorse plugin. Other applications, like Evolution, support encrypt directly.

The first time you use **gpg**, a **.gnupg** directory is created in your home directory with a file named **options**. The **.gnupg/gpg.conf** file contains commented default options for GPG operations. You can edit this file and uncomment or change any default options you want implemented for GPG. You can use a different options file by specifying it with the **--options** parameter when invoking **gpg**. Helpful options include keyserver entries. The **.gnupg** directory will also hold encryption files such as **secring.gpg** for your secret keys (secret keyring), **pubring.gpg** for your public keys (public keyring), and **trustdb.gpg**, which is a database for trusted keys.

Generating your public key with gpg

Before you can use GnuPG, you will have to generate your private and public keys. You can do this with the Passwords and Encryption Keys utility as described in the previous section, or with the **gpg** command entered in a terminal window, as described here. On the command line (terminal window), enter the **gpg** command with the **--gen-key** command. The **gpg** program will then prompt with different options for creating your private and public keys. You can check the **gpg** Man page for information on using the **gpg** program.

```
gpg --gen-key
```

GPG Commands	Description
-s, --sign	Signs a document, creating a signature. May be combined with **--encrypt**.
--clearsign	Creates a clear-text signature.
-b, --detach-sign	Creates a detached signature.
-e, --encrypt	Encrypts data. May be combined with **--sign**.
--decrypt [*file*]	Decrypts file (or **stdin** if no file is specified) and writes it to **stdout** (or the file specified with **--output**). If the decrypted file is signed, the signature is verified.
--verify [[*sigfile*] [*signed-files*]]	Verifies a signed file. The signature can be either contained with the file or a separate detached signature file.
-k, --list-keys [*names*], **--list-public-keys** [*names*]	Lists all keys from the public keyrings or those specified.
-K, --list-secret-keys [*names*]	Lists your private (secret) keys.
--list-sigs [*names*]	Lists your keys along with any signatures they have.
--check-sigs [*names*]	Lists keys and their signatures and verifies the signatures.
--fingerprint [*names*]	Lists fingerprints for specified keys.
--gen-key	Generates a new set of private and public keys.
--edit-key *name*	Edits your keys. Commands perform most key operations, such as **sign** to sign a key or **passwd** to change your passphrase.
--sign-key *name*	Signs a public key with your private key. Same as **sign in --edit-key**.
--delete-key *name*	Removes a public key from the public keyring.
--delete-secret-key *name*	Removes private and public keys from both the secret and public keyrings.
--gen-revoke	Generates a revocation certificate for your own key.
--export [*names*]	Exports a specified key from your keyring. With no arguments, exports all keys.
--send-keys [*names*]	Exports and sends specified keys to a keyserver. The option **--keyserver** must be used to give the name of this keyserver.
--import [*files*]	Imports keys contained in files into your public keyring.
GPG Options	**Description**

-a, --armor	Creates ASCII armored output, ASCII version of encrypted data.
-o, --output *file*	Writes output to a specified file.
--default-key *name*	Specifies the default private key to use for signatures.
--keyserver *site*	Looks up public keys not on your keyring.
-r, --recipient *names*	Encrypts data for the specified user, using that user's public key.
--default-recipient *names*	Specifies the default recipient to use for encrypting data.

Table 10-2: GPG Commands and Options

You are first asked to select the kind of key you want. Normally, you just select the default entry, which you can do by pressing the ENTER key.

Then you choose the key size, usually the default 1024.

You then specify how long the key is to be valid—usually, there is no expiration.

You are then asked to enter a user ID, a comment, and an e-mail address. Press ENTER to be prompted for each in turn. These elements, any of which can be used as the key's name, identify the key. You use the key name when performing certain GPG tasks such as signing a key or creating a revocation certificate. For example, the following elements create a key for the user richlp with the comment "author" and the e-mail address **richlp@turtle.mytrek.com**:

```
Richard Petersen (author) <richlp@turtle.mytrek.com>
```

You can use any unique part of a key's identity to reference that key. For example, the string "Richard" would reference the preceding key, provided there are no other keys that have the string "Richard" in them. The string "richlp" would also reference the key, as would "author". Where a string matches more than one key, all those matched would be referenced.

After you have entered your user ID, comment, and email address, the elements are displayed along with a menu is which will allow you change any element.

```
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit?
```

Enter **o** to approve and accept the key.

The **gpg** program will then ask you to enter a passphrase, used to protect your private key. Be sure to use a real phrase, including spaces, not just a password.

gpg then generates your public and private keys and places them in the **.gnupg** directory. This may take a few minutes.

The private keys are kept in a file called **secring.gpg** in your **.gnupg** directory. The public key is placed in the **pubring.gpg** file, to which you can add the public keys of other users. You can list these keys with the **--list-keys** and **--list-public-keys** commands.

```
gpg --list-keys
```

To list your private keys you would use the **--list-secret-keys** command.

```
gpg --list-secret-keys
```

If you need to change your keys later, you can create a revocation certificate to notify others that the public key is no longer valid. For example, if you forget your password or someone else discovers it, you can use the revocation certificate to tell others that your public key should no longer be used. In the next example, the user creates a revocation certificate for the key richlp and places it in the file **myrevoke.asc**:

```
gpg --output myrevoke.asc --gen-revoke richlp
```

Importing Public Keys

To decode messages from other users, you will need to have their public keys. Either they can send them to you or you can download them from a keyserver. Save the message or web page containing the public key to a file. You will then need to import the key, and you should also verify and sign the key. Use the file you received to import the public key to your **pubring** file. As noted previously, you can also use the Seahorse Passwords and Encryptions Keys utility (Applications | Accessories | Passwords and Encryption Keys) to import and sign keys. In the following example, the user imports George's public key, which he has received as the file **georgekey.asc**.

```
gpg --import georgekey.asc
```

You can also use the **gpg --search-key** and **--keyserver** options to import the key. Keys matching the search term will be displayed in a numbered list. You will be prompted to enter the number of the key you want. The 2007 Sendmail key from the results from the following example would be 7. This is the key used for 2007 released software.

```
$ gpg  --keyserver pgp.mit.edu  --search-keys Sendmail
gpg: searching for "Sendmail" from hkp server pgp.mit.edu
(1)     Sendmail Signing Key/2008 <sendmail@Sendmail.ORG>
          1024 bit RSA key F6B30729, created: 2008-01-18
. . . . . .
(7)     Sendmail Signing Key/2007 <sendmail@Sendmail.ORG>
          1024 bit RSA key 7093B841, created: 2006-12-16
Enter number(s), N)ext, or Q)uit > 7
gpg: requesting key 7093B841 from hkp server pgp.mit.edu
gpg: key 7093B841: public key "Sendmail Signing Key/2007 <sendmail@Sendmail.ORG>"
imported
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

Note: You can remove any key, including your own private key, with the **--delete-key** and **--delete-secret-key** commands.

Signing Your Public Keys

If you trust the imported key, you can then sign it, making it a trusted key (these will show up in the Trusted Keys tab of the Passwords and Encryption Keys window). To sign a key, you use the **gpg** command with the **--sign-key** command and the key's name.

```
gpg --sign-key george@rabbit
```

Alternatively, you can edit the key with the **--edit-key** command to start an interactive session in which you can enter the command **sign** to sign the key and **save** to save the change. Signing a key involves accessing your private key, so you will be prompted for your passphrase. When you are finished, leave the interactive session with the **quit** command.

In this example, the e-mail address is used for the key name. You are prompted to make sure you want to sign it. Then you have to enter the passphrase for your own GPG key.

```
$ gpg --sign-key sendmail@Sendmail.ORG
pub  1024R/7093B841  created: 2006-12-16  expires: never       usage: SCEA
                     trust: unknown       validity: unknown
[ unknown] (1). Sendmail Signing Key/2007 <sendmail@Sendmail.ORG>
pub  1024R/7093B841  created: 2006-12-16  expires: never       usage: SCEA
                     trust: unknown       validity: unknown
 Primary key fingerprint: D9 FD C5 6B EE 1E 7A A8  CE 27 D9 B9 55 8B 56 B6
     Sendmail Signing Key/2007 <sendmail@Sendmail.ORG>
Are you sure that you want to sign this key with your
key "Richard Petersen <richard@somedomain>" (0108D72C)
Really sign? (y/N) y
You need a passphrase to unlock the secret key for
user: "Richard Petersen <richard@somedomain>"
1024-bit DSA key, ID 0108D72C, created 2008-03-26
```

For public keys in your keyrings, you can set different trust levels. GPG supports several trust levels, including marginal, full trust, and ultimate. You use the **--edit-key** command with the **trust** option to set the trust level.

```
gpg --edit-key george@rabbit trust
```

This will display a menu of several options

```
1 = I don't know or won't say
2 = I do NOT trust
3 = I trust marginally
4 = I trust full
5 = I trust ultimately
m = back to main menu
```

The **--edit-key** command actually runs a shell in which you can enter a variety of different key modification operations, like **trust** to set the trust level, **keyserver** to tell where the key can be found, and **sign** to sign the key. Use the **quit** command to leave the edit-key shell.

You can also check the fingerprint of the key for added verification. To manually check that a public key file was not modified in transit, you can check its fingerprint. This is a hash value generated from the contents of the key, much like a modification digest. Using the **--fingerprint** option, you can generate a hash value from the key you installed, and then contact the sender and ask them what the hash value should really be. If they are not the same, you know the key was tampered with in transit.

Tip: You can use the **--fingerprint** option to check a key's validity if you wish. If you are confident that the key is valid, you can then sign it with the **--sign-key** command.

You do not have to check the fingerprint to have **gpg** operate. This is just an advisable precaution you can perform on your own. The point is that you need to be confident that the key

you received is valid. Normally you can accept most keys from public servers or known sites as valid, though it is easy to check their posted fingerprints. Once assured of the key's validity, you can then sign it with your private key. Signing a key notifies **gpg** that you officially accept the key.

Publishing keys

You can use the **gpg** command with the **-keyserver** option and **--send-key** command to send keys directly to a keyserver. The **--send-key** command takes as its argument your e-mail address. You need to send to only one keyserver, as it will share your key with other keyservers automatically.

```
gpg --keyserver pgp.mit.edu:11371 --send-key chris@turtle.mytrek.com
```

If you want to send your key directly to another user, you should generate an armored text version of the key that you can then e-mail. You do this with the **--armor** and **--export** options, using the **--output** option to specify a file to place the key in. The **--armor** option will generate an ASCII text version of the encrypted file so that it can be e-mailed directly, instead of as an attached binary. Files that hold an ASCII-encoded version of the encryption normally have the extension **.asc**, by convention. Binary encrypted files normally use the extension **.gpg**. You can then e-mail the file to users you want to send encrypted messages.

```
# gpg --armor --export richlp@turtle.mytrek.com --output richlp.asc
# mail -s 'mypubkey' george@rabbit.mytrek.com < richlp.asc
```

Many companies and institutions post their public key files on their websites, where they can be downloaded and used to verify encrypted software downloads or official announcements.

Note: Some commands and options for GPG have both long and short forms. For example, the **--armor** command can be written as **-a**, **--output** as **-o**, **--sign** as **-s**, and **--encrypt** as **-e**. Most others, like **--export**, have no short form.

Using GnuPG

GnuPG encryption is currently supported by most mail clients, including Kmail, Thunderbird, and Evolution. You can use the **gpg** command to manually encode and decode messages, including digital signatures. Seahorse provides several GPG encryption plugins for use with Evolution and Gedit.

Encrypting and decrypting data with gpg command

The **gpg** command provides several options for managing secure messages. The **e** option encrypts messages, the **a** option generates an armored text version, and the **s** option adds a digital signature. Email applications will use this option. You will need to specify the recipient's public key, which you should already have imported into your **pubring** file. It is this key that is used to encrypt the message. The recipient will then be able to decode the message with their private key. Use the **--recipient** or **-r** option to specify the name of the recipient key. You can use any unique substring in the user's public key name. The e-mail address usually suffices. You use the **d** option to decode received messages. In the following example, the user encrypts (**e**) and signs (**s**) a file generated in armored text format (**a**). The **-r** option indicates the recipient for the message (whose public key is used to encrypt the message).

```
gpg e -s -a -o myfile.asc -r george@rabbit.mytrek.com myfile
# mail george@rabbit.mytrek.com < myfile.asc
```

You can leave out the ASCII armor option if you want to send or transfer the file as a binary attachment. Without the **--armor** or **-a** option, **gpg** generates an encoded binary file, not an encoded text file. This is the method used for encryption by Nautilus. A binary file can be transmitted through e-mail only as an attachment. As noted previously, ASCII armored versions usually have an extension of **.asc**, whereas binary version use **.gpg**.

When the other user receives the file, they can save it to a file named something like **myfile.asc** and then decode the file with the **-d** option. The **-o** option will specify a file to save the decoded version in. GPG will automatically determine if it is a binary file or an ASCII armored version.

```
gpg -d -o myfile.txt myfile.asc
```

To check the digital signature of the file, you use the **gpg** command with the **--verify** option. This assumes that the sender has signed the file.

```
gpg --verify myfile.asc
```

Note: You can use **gpgsplit** to split a GPG message into its components to examine them separately.

As you perform GPG tasks, you will need to reference the keys you have using their key names. Bear in mind that you need only a unique identifying substring to select the key you want. GPG performs a pattern search on the string you specify as the key name in any given command. If the string matches more than one key, all those matching will be selected.

Seahorse plugins: Choose Recipients

Plug-ins are provided for Gedit editor to encrypt text files, the Epiphany Web browser for text phrases, and Nautilus to perform encryption from the context menu. A panel applet lets you encrypt, sign, and decrypt clipboard content.

The Seahorse plugin opens a Choose Recipients window in which you can choose the key to use for encryption (see Figure 10-10). A pop-up menu lets you use all keys, just selected recipients, or search results. A search box lets you search for keys, selecting them based on a pattern you enter. Available keys will be listed in the window by name and key ID. You also have the option to sign the message, selecting signatures from the Sign Message As pop-up menu. Once you make your selection, you will be prompted to enter the passphrase for that encryption key.

Encrypting and decrypting files with Nautilus

Nautilus will generate an encrypted copy of a file, giving that copy the extension **.gpg**. It operates like gpg with just the **-e** option, and no **-a** option. To encrypt a file from Nautilus, select the file and then right-click to open the Nautilus pop-up menu. On this menu select the Encrypt option. You can also select the file and then choose Encrypt from the Edit menu. The Choose Recipients window then opens letting you select the encryption keys and digital signature to use. Select the encryption key. You will be prompted to enter the key's passphrase. Then an encrypted copy of the file will be generated with the extension **.pgp**. The original is left untouched.

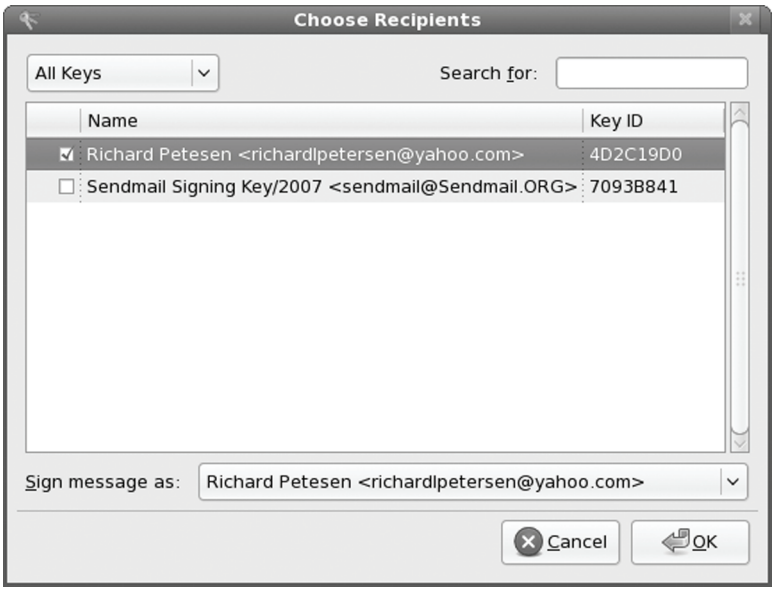


Figure 10-10: Choose Recipients window.

If you just want to sign a file, you can select the Sign entry in the Edit menu (or right-click on the filename/icon and choose Sign from the pop-up menu). This opens a dialog with a pop-up menu listing digital signatures you can use.

To decrypt the encrypted .pgp file, simply double-click on it or right-click and choose Open With to select Decrypt File. This opens the file with the decrypt tool which will generate a decrypted copy of the file. A "Choose decrypted file name" dialog will then open where you can enter the name for the copy and the directory to save it in. You are then prompted for the passphrase.

Encrypting data with Gedit

Gedit is designed to create armored text encrypted files, the kind you would send as an email. It will change the original text file, transforming the text into an encoded armor ASCII equivalent, with BEGIN and END entries for the encoded data. You could then send the text directly as a message. To decode, be sure to select the entire encoded text, including the BEGIN and END lines. You will be prompted for the passphrase for the key. If signed, the signature will also be checked.

To encrypt files with Gedit, you first have to enable encryption. Open Gedit and select preferences from the Edit menu. On the Preferences window select the Plugins tab. Scroll down the list of active plugins ad click the checkbox for Text Encryption. Now, on the Gedit Edit menu, you will see entries for Sign, Decrypt/Verify, and Encrypt. Choose Encrypt to encrypt the message, Sign to just sign it. When you choose Encrypt, the Choose Recipients dialog opens where you can select the encryption keys to use. If you choose Sign, a small dialog appears with a pop-up menu listing digital signatures you can use. To decrypt or verify, first select the text and then select the Decrypt/Verify entry.

Decrypting a Digital Signature

You will need to have the signer's public key to decode and check the digital signature. If you do not have the key, you will receive a message saying that the public key was not found. In this case, you will first have to obtain the signer's public key. You can access a keyserver that you think may have the public key or request the public key directly from a website or from the signer. Then import the key as described earlier.

Signing Messages

Most applications that handle text and data files provide the ability to sign those files. For the Gedit editor and Nautilus file browser, a window opens with a pop-up menu letting you choose the key to use to sign the file. On Evolution you can select the PGP signature entry from the Security menu in the Compose message window.

Note: One very effective use for digital signatures is to verify that a software package has not been altered. A software package could be intercepted in transmission and some of its system-level files changed or substituted. Software packages for Fedora, as well as those by reputable GNU and Linux projects, are digitally signed. The signature provides modification digest information with which to check the integrity of the package.

You do not have to encrypt a file to sign it. A digital signature is a separate component. You can either combine the signature with a given file or generate one separately. To combine a signature with a file, you generate a new version that incorporates both. Use the **--sign** or **-s** option to generate a version of the document that includes the digital signature. In the following example, the **mydoc** file is digitally signed with **mydoc.gpg** file containing both the original file and the signature.

```
gpg  -o mydoc.gpg  --sign mydoc
```

If, instead, you want to just generate a separate signature file, you use the **--detach-sig** command. This has the advantage of not having to generate a complete copy of the original file. That file remains untouched. The signature file usually has an extension such as **.sig**. In the following example, the user creates a signature file called **mydoc2.sig** for the **mydoc2** file.

```
gpg -o mydoc2.sig --detach-sig mydoc2
```

To verify the file using a detached signature, the recipient user specifies both the signature file and the original file.

```
gpg --verify mydoc2.sig  mydoc2
```

To verify a trusted signature you can use **gpgv**.

You can also generate a clear sign signature to be used in text files. A *clear sign* signature is a text version of the signature that can be attached to a text file. The text file can be further edited by any text editor. Use the **--clearsign** option to create a clear sign signature. The following example creates a clear signed version of a text file called **mysignotice.txt**.

```
gpg -o mysignotice.txt --clearsign mynotice.txt
```

fedora

11. File and Directory Access

Permissions: Discretionary Access Control
Access Control Lists: FACL
Encrypted File Systems
Intrusion Detection: Sectool

Access to files and directories can be directly controlled by permissions and access controls. Permissions are the traditional method for controlling access. They limit access by a broad set of restrictions for owner, group, or anyone else. For more refined access, like specifying access by particular users, you can use Access Control Lists (ACL). Permissions are common used on most Unix and Linux systems and remain the simplest way to control access. ACLs require more maintenance, but offer many more access options.

Another level of control can be applied to file systems using file system encryption. Administrators can block any access to a file system, except to authorized users. In effect, encryption implements password protected file system.

Permissions: Discretionary Access Control

Each file and directory in Linux contains a set of permissions that determine who can access it and how. These are known as Discretionary Access Controls (DACs). You set these permissions to limit access in one of three ways: you can restrict access to yourself alone, you can allow users in a pre-designated group to have access, or you can permit anyone on your system to have access. You can also control how a given file or directory is accessed.

read, write, and execute

A file or directory may have read, write, and execute permissions. When you create a file, you, as the creator and owner, are automatically given read and write permissions for the owner, enabling you to display and modify the file. You may change these permissions to any combination you want. A file can also have read-only permission, which prevents any modifications.

Tip: From GNOME and KDE desktops you can change permissions easily by right-clicking on a file or directory icon and selecting Properties. On the Properties window's Permissions tab you will see options for setting Owner, Group, and Other permissions.

Three different categories of users can have access to a file or directory: the owner, the group, and all others not belonging to that group. The owner is the user who created the file. Any file you create, you own. You can also permit a group to have access to a file. Users are often collected into groups. For example, all the users for a given class or project can be formed into a group by the system administrator. A user can grant access to a file to the members of a designated group. Finally, you can also open up access to a file to all other users on the system. In this case, every user not part of the file's group can have access to that file. In this sense, every other user on the system makes up the "others" category. If you want to give the same access to all users on your system, you set the same permissions for both the group and the others. That way, you include both members of the group (group permission) and all those users who are not members (others permission).

Each category has its own set of read, write, and execute permissions. The first set controls the user's own access to his or her files—the owner access. The second set controls the access of the group to a user's files. The third set controls the access of all other users to the user's files. The three sets of read, write, and execute permissions for the three categories—owner, group, and other—make a total of nine types of permissions.

The **ls** command with the **-l** option displays detailed information about the file, including the permissions. In the following example's second line, the first few set of characters on the left is a list of the permissions set for the **mydata** file:

```
$ ls -l mydata
-rw-r--r-- 1 chris weather 207 Feb 20 11:55 mydata
```

An empty permission is represented by a dash, **-**. The read permission is represented by **r**, write by **w**, and execute by **x**. There are ten positions for ten characters. The first character indicates the file type. In a general sense, a directory can be considered a type of file. If the first character is a dash, it means a file is being listed. If the first character is **d**, information about a directory is being displayed.

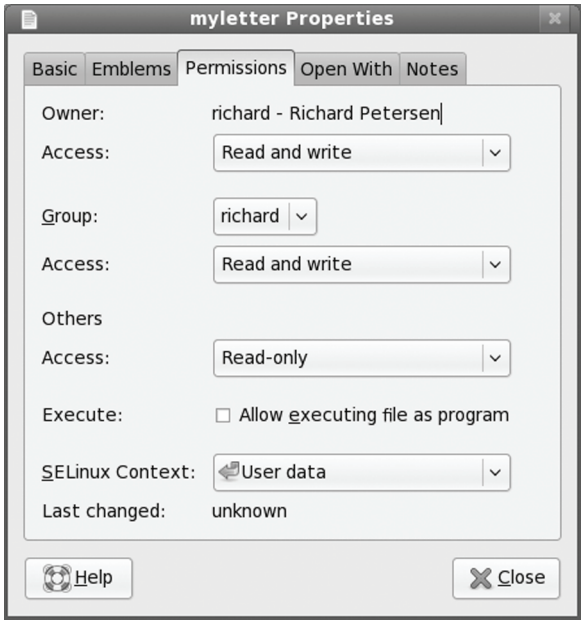


Figure 11-1: File Permissions on GNOME

The next nine characters are arranged according to the different user categories. The first set of three characters is the owner's set of permissions for the file. The second set of three characters is the group's set of permissions for the file. The last set of three characters is the other users' set of permissions for the file.

Permissions on GNOME and KDE

On GNOME, you can set a directory or file permission using the Permissions tab in its Properties window. Right-click the file or directory entry in the file manager window and select Properties. Then select the Permissions tab. Here you will find pop-up menus for Read, Write, and Execute along with rows for Owner, Group, and Other. You can set owner permissions as Read Only or Read And Write. For the group and others, you can also set the None option, denying

access. The group name expands to a pop-up menu listing different groups; select one to change the file's group. If you want to execute this as an application (say, a shell script) check the Allow Executing File As Program entry. This has the effect of setting the execute permission. For SELinux you can set the security context using the pop-up menu. The user data security context will be selected by default. Users will also have the option to choose the Temporary data context, as well as others that may be enabled for file.

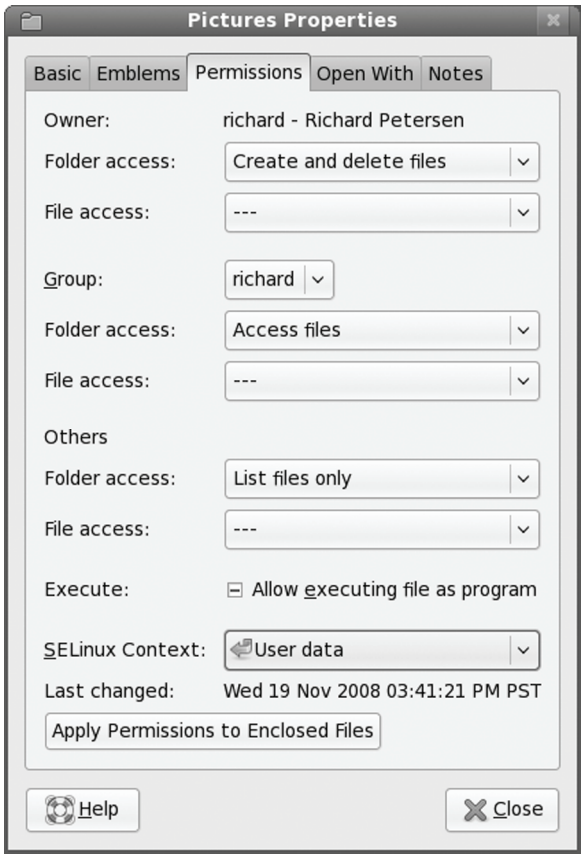


Figure 11-2: Directory Permissions on GNOME

The Permissions tab for directories operates much the same way, but it includes two access entries, Folder Access and File Access. The Folder Access entry controls access to the folder with options for List Files Only, Access Files, and Create And Delete Files. These correspond to the read, read and execute, and read/write/execute permissions granted to directories. The File Access entry lets you set permissions for all those files in the directory. They are the same as for files: for the owner, Read or Read and Write; for the group and others, the entry adds a None option to deny access. To set the permissions for all the files in the directory accordingly (not just the folder), click the Apply Permissions To Enclosed Files button.

For SELinux you can set the security context using the pop-up menu. The user data security context will be selected by default. Shared directories for Samba will also have a Samba (cifs) security context.

If you are not the owner of a directory or file, you will not be able to change any of the permissions. The entries will be grayed out, but with the current permissions displayed.

Permissions on KDE

On KDE, you can set a directory or file permission using the Permissions tab in its Properties window. Right-click the file or directory entry in the file manager window and select Properties. Then select the Permissions tab. Here you will find pop-up menus for Owner, Group, and Others. Options include Can Read, Can Read and Write, and Forbidden. For more refined access, click on the Advanced Permissions button to display a table for checking read, write, and execute access (**r**, **w**, **x**) for owner, group, and others. You can also set the sticky bit and user and group ID permissions. The Add Entry button lets you set up ACL access, specifying certain users or groups that can or cannot have access to the file.

Directories have slightly different options: Can View Content and Can View and Modify Content which are the read and write permissions. You have the option to apply changes to all subdirectories and the files in them. Clicking on the Advanced Permissions button displays the same read, write, and execute table for owner, group, and others. Click a table entry to toggle a permission on or off. The selected permissions are shown in the Effective column. Use the Add Entry button to add ACL entries to control access by specific users and groups.

chmod

You use the **chmod** command to change different permission configurations. **chmod** takes two lists as its arguments: permission changes and filenames. You can specify the list of permissions in two different ways. One way uses permission symbols and is referred to as the *symbolic method.* The other uses what is known as a "binary mask" and is referred to as either the absolute or the relative method. Table 11-1 lists options for the **chmod** command.

Ownership

Files and directories belong to both an owner and a group. A group usually consists of a collection of users, all belonging to the same group. In the following example, the **mydata** file is owned by the user **robert** and belongs to the group **weather**:

```
-rw-r--r-- 1 robert weather 207 Feb 20 11:55 mydata
```

A group can also consist of one user, normally the user who creates the file. Each user on the system, including the root user, is assigned his or her own group of which he or she is the only member, ensuring access only by that user. In the next example, the report file is owned by the **robert** user and belongs to that user's single user group, **robert**:

```
-rw-r--r-- 1 robert robert 305 Mar 17 12:01 report
```

The root user, the system administrator, owns most of the system files that also belong to the root group, of which only the root user is a member. Most administration files, like configuration files in the **/etc** directory, are owned by the root user and belong to the root group. Only the root user has permission to modify them, whereas normal users can read and, in the case

of programs, also execute them. In the next example, the root user owns the **fstab** file in the **/etc** directory, which also belongs to the root user group.

```
-rw-r--r-- 1 root root 621 Apr 22 11:03 fstab
```

Command or Option	Execution
chmod	Changes the permission of a file or directory.
Options	
+	Adds a permission.
-	Removes a permission.
=	Assigns entire set of permissions.
r	Sets read permission for a file or directory. A file can be displayed or printed. A directory can have the list of its files displayed.
w	Sets write permission for a file or directory. A file can be edited or erased. A directory can be removed.
x	Sets execute permission for a file or directory. If the file is a shell script, it can be executed as a program. A directory can be changed to and entered.
u	Sets permissions for the user who created and owns the file or directory.
g	Sets permissions for group access to a file or directory.
o	Sets permissions for access to a file or directory by all other users on the system.
a	Sets permissions for access by the owner, group, and all other users.
s	Sets User ID and Group ID permission; program owned by owner and group.
t	Sets sticky bit permission; program remains in memory.
Commands	
chgrp *groupname filenames*	Changes the group for a file or files.
chown *user-name filenames*	Changes the owner of a file or files.
ls -l *filename*	Lists a filename with its permissions displayed.
ls -ld *directory*	Lists a directory name with its permissions displayed.
ls -l	Lists all files in a directory with its permissions displayed.

Table 11-1: File and Directory Permission Operations

Certain directories and files located in the system directories are owned by a service, rather than the root user, because the services need to change those files directly. This is

particularly true for services that interact with remote users, such as Internet servers. Most of these files are located in the **/var** directory. Here you will find files and directories managed by services like the Squid proxy server and the Domain Name Server (named). In this example, the Squid proxy server directory is owned by the **squid** user and belongs to the **squid** group:

```
drwxr-x--- 2 squid squid 4096 Jan 24 16:29 squid
```

Note: When a program is owned by the root, setting the user ID permission will give the user the ability to execute the program with root permissions. This can be a serious security risk for any program that can effect changes—such as **rm**, which removes files.

Changing a File's Owner or Group: chown and chgrp

Although other users may be able to access a file, only the owner can change its permissions. If, however, you want to give some other user control over one of your file's permissions, you can change the owner of the file from yourself to the other user. The **chown** command transfers control over a file to another user. This command takes as its first argument the name of the other user. Following the username, you list the files you are transferring. In the next example, the user gives control of the **mydata** file to user **robert**:

```
$ chown robert mydata
$ ls -l mydata
-rw-r--r-- 1 robert weather 207 Feb 20 11:55 mydata
```

You can also, if you wish, change the group for a file, using the **chgrp** command. **chgrp** takes as its first argument the name of the new group for a file or files. Following the new group name, you list the files you want changed to that group. In the next example, the user changes the group name for **today** and **weekend** to the **forecast** group. The **ls -l** command then reflects the group change.

```
$ chgrp forecast today weekend
$ ls -l
-rw-rw-r-- 1 chris forecast 568 Feb 14 10:30 today
-rw-rw-r-- 1 chris forecast 308 Feb 17 12:40 weekend
```

You can combine the **chgrp** operation with the **chown** command by attaching a group to the new owner with a colon.

```
$ chown george:forecast tomorrow
-rw-rw-r-- 1 george forecast 568 Feb 14 10:30 tomorrow
```

Setting Permissions: Permission Symbols

The symbolic method of setting permissions uses the characters **r**, **w**, and **x** for read, write, and execute, respectively. Any of these permissions can be added or removed. The symbol to add a permission is the plus sign, **+**. The symbol to remove a permission is the minus sign, **-**. In the next example, the **chmod** command adds the execute permission and removes the write permission for the **mydata** file for all categories. The read permission is not changed.

```
$ chmod +x-w mydata
```

Permission symbols also specify each user category. The owner, group, and others categories are represented by the **u**, **g**, and **o** characters, respectively. Notice the owner category is

represented by a **u** and can be thought of as the user. The symbol for a category is placed before plus and minus sign preceding the read, write, and execute permissions. If no category symbol is used, all categories are assumed, and the permissions specified are set for the user, group, and others. In the next example, the first **chmod** command sets the permissions for the group to read and write. The second **chmod** command sets permissions for other users to read. Notice no spaces are between the permission specifications and the category. The permissions list is simply one long phrase, with no spaces.

```
$ chmod g+rw mydata
$ chmod o+r mydata
```

A user may remove permissions as well as add them. In the next example, the read permission is set for other users, but the write and execute permissions are removed:

```
$ chmod o+r-wx mydata
```

Another permission character exists, **a**, which represents all the categories. The **a** character is the default. In the next example, the two commands are equivalent. The read permission is explicitly set with the **a** character denoting all types of users: other, group, and user.

```
$ chmod a+r mydata
$ chmod +r mydata
```

One of the most common permission operations is setting a file's executable permission. This is often done in the case of shell program files. The executable permission indicates a file contains executable instructions and can be directly run by the system. In the next example, the file **lsc** has its executable permission set and then executed:

```
$ chmod u+x lsc
$ lsc
main.c lib.c
$
```

Absolute Permissions: Binary Masks

Instead of the permission symbols in Table 11-1, many users find it more convenient to use the absolute method. The *absolute method* changes all the permissions at once, instead of specifying one or the other. It uses a *binary mask* that references all the permissions in each category. The three categories, each with three permissions, conform to an octal binary format. Octal numbers have a base 8 structure. When translated into a binary number, each octal digit becomes three binary digits. A binary number is a set of 1 and 0 digits. Three octal digits in a number translate into three sets of three binary digits, which is nine altogether—and the exact number of permissions for a file.

You can use the octal digits as a mask to set the different file permissions. Each octal digit applies to one of the user categories. You can think of the digits matching up with the permission categories from left to right, beginning with the owner category. The first octal digit applies to the owner category, the second to the group, and the third to the others category. The actual octal digit you choose determines the read, write, and execute permissions for each category. At this point, you need to know how octal digits translate into their binary equivalents.

Calculating Octal Numbers

A simple way to calculate the octal number makes use of the fact that any number used for permissions will be a combination derived from adding in decimal terms the numbers 4, 2, and 1. Use 4 for read permission, 2 for write, and 1 for execute. The read, write, execute permission is simply the addition of 4 + 2 + 1 to get 7. The read and execute permission adds 4 and 1 to get 5. You can use this method to calculate the octal number for each category. To get 755, you would add 4 + 2 + 1 for the owner read, write, and execute permission, 4 + 1 for the group read and execute permission, and 4 + 1 again for the other read and execute permission.

Binary Masks

When dealing with a binary mask, you need to specify three digits for all three categories, as well as their permissions. This makes a binary mask less versatile than the permission symbols. To set the owner execute permission on and the write permission off for the **mydata** file and retain the read permission, you need to use the octal digit 5 (101). At the same time, you need to specify the digits for group and other user's access. If these categories are to retain read access, you need the octal number 4 for each (100). This gives you three octal digits, 544, which translate into the binary digits 101 100 100.

```
$ chmod 544 mydata
```

Execute Permissions

One of the most common uses of the binary mask is to set the execute permission. You can create files that contain Linux commands, called *shell scripts.* To execute the commands in a shell script, you must first indicate the file is executable—that it contains commands the system can execute. You can do this in several ways, one of which is to set the executable permission on the shell script file. Suppose you just completed a shell script file and you need to give it executable permission to run it. You also want to retain read and write permission but deny any access by the group or other users. The octal digit 7 (111) will set all three permissions, including execute (you can also add 4-read, 2-write, and 1-execute to get 7). Using 0 for the group and other users denies them access. This gives you the digits 700, which are equivalent to the binary digits 111 000 000. In this example, the owner permission for the **myprog** file is set to include execute permission:

```
$ chmod 700 myprog
```

If you want others to be able to execute and read the file but not change it, you can set the read and execute permissions and turn off the write permission with the digit 5 (101). In this case, you use the octal digits 755, having the binary equivalent of 111 101 101.

```
$ chmod 755 myprog
```

Directory Permissions

You can also set permissions on directories. The read permission set on a directory allows the list of files in a directory to be displayed. The execute permission enables a user to change to that directory. The write permission enables a user to create and remove his or her files in that directory. If you allow other users to have write permission on a directory, they can add their own files to it. When you create a directory, it is automatically given read, write, and execute permission for the owner. You may list the files in that directory, change to it, and create files in it.

Like files, directories have sets of permissions for the owner, the group, and all other users. Often, you may want to allow other users to change to and list the files in one of your directories but not let them add their own files to it. In this case, you set read and execute permissions on the directory but you don't set a write permission. This allows other users to change to the directory and list the files in it but not to create new files or to copy any of their files into it. The next example sets read and execute permissions for the group for the **thankyou** directory but removes the write permission. Members of the group may enter the **thankyou** directory and list the files there, but they may not create new ones.

```
$ chmod g+rx-w letters/thankyou
```

Just as with files, you can also use octal digits to set a directory permission. To set the same permissions as in the preceding example, you use the octal digits 750, which have the binary equivalents of 111 101 000.

```
$ chmod 750 letters/thankyou
```

Displaying Directory Permissions

The **ls** command with the **-l** option lists all files in a directory. To list only the information about the directory itself, add a **d** modifier. In the next example, **ls -ld** displays information about the **thankyou** directory. Notice the first character in the permissions list is **d**, indicating it is a directory:

```
$ ls -ld thankyou
drwxr-x--- 2 chris 512 Feb 10 04:30 thankyou
```

Parent Directory Permissions

If you have a file you want other users to have access to, you not only need to set permissions for that file, you also must make sure the permissions are set for the directory in which the file is located. To access your file, a user must first access the file's directory. The same applies to parents of directories. Although a directory may give permission to others to access it, if its parent directory denies access, the directory cannot be reached. Therefore, you must pay close attention to your directory tree. To provide access to a directory, all other directories above it in the directory tree must also be accessible to other users.

Ownership Permissions

In addition to the read/write/execute permissions, you can also set ownership permissions for executable programs. Normally, the user who runs a program owns it while it is running, even though the program file itself may be owned by another user. The Set User ID permission allows the original owner of the program to own it always, even while another user is running the program. For example, most software on the system is owned by the root user but is run by ordinary users. Some such software may have to modify files owned by the root. In this case, the ordinary user needs to run that program with the root retaining ownership so that the program can have the permissions to change those root-owned files. The Group ID permission works the same way, except for groups. Programs owned by a group retain ownership, even when run by users from another group. The program can then change the owner group's files. There is a potential security risk involved in that you are essentially giving a user some limited root-level access.

Ownership Permissions Using Symbols

To add both the User ID and Group ID permissions to a file, you use the **s** option. The following example adds the User ID permission to the **pppd** program, which is owned by the root user. When an ordinary user runs **pppd**, the root user retains ownership, allowing the **pppd** program to change root-owned files.

```
# chmod +s /usr/sbin/pppd
```

The Set User ID and Set Group ID permissions show up as an **s** in the execute position of the owner and group segments. Set User ID and Group ID are essentially variations of the execute permission, **x**. Read, write, and User ID permission are **rws** instead of just **rwx**.

```
# ls -l /usr/sbin/pppd
-rwsr-sr-x 1 root root 184412 Jan 24 22:48 /usr/sbin/pppd
```

Ownership Permissions Using the Binary Method

For the ownership permissions, you add another octal number to the beginning of the octal digits. The octal digit for User ID permission is 4 (100) and for Group ID, it is 2 (010) (use 6 to set both—110). The following example sets the User ID permission to the **pppd** program, along with read and execute permissions for the owner, group, and others:

```
# chmod 4555 /usr/sbin/pppd
```

Sticky Bit Permissions

One other special permission provides for greater security on directories, the *sticky bit.* Originally the sticky bit was used to keep a program in memory after it finished execution to increase efficiency. Current Linux systems ignore this feature. Instead, it is used for directories to protect files within them. Files in a directory with the sticky bit set can only be deleted or renamed by the root user or the owner of the directory.

Sticky Bit Permission Using Symbols

The sticky bit permission symbol is **t**. The sticky bit shows up as a **t** in the execute position of the other permissions. A program with read and execute permissions with the sticky bit has its permissions displayed as **r-t**.

```
# chmod +t /home/dylan/myreports
# ls -l /home/dylan/myreports
-rwxr-xr-t 1 root root 4096 /home/dylan/myreports
```

Sticky Bit Permission Using the Binary Method

As with ownership, for sticky bit permissions, you add another octal number to the beginning of the octal digits. The octal digit for the sticky bit is 1 (001). The following example sets the sticky bit for the **myreports** directory:

```
# chmod 1755 /home/dylan/myreports
```

The next example sets both the sticky bit and the User ID permission on the **newprogs** directory. The permission 5755 has the binary equivalent of 101 111 101 101:

```
# chmod 5755 /usr/bin/newprogs
# ls -l /usr/bin/newprogs
drwsr-xr-t 1 root root 4096  /usr/bin/newprogs
```

Permission Defaults: umask

Whenever you create a file or directory, it is given default permissions. You can display the current defaults or change them with the **umask** command. The permissions are displayed in binary or symbolic format as described in the following sections. The default permissions include any execute permissions that are applied to a directory. Execute permission for a file is turned off by default when you create it because standard data files do not use the executable permissions (to make a file executable like a script, you have to manually set it's execute permission). To display the current default permissions, use the **umask** command with no arguments. The **-S** option uses the symbolic format.

```
$ umask -S
u=rwx,g=rx,o=rx
```

This default umask provides **rw-r--r--** permission for standard files and adds execute permission for directories, **rwxr-xr-x**.

You can set a new default by specifying permissions in either symbolic or binary format. To specify the new permissions, use the **-S** option. The following example denies others read permission, while allowing user and group read access, which results in permissions of **rwxr-x---:**

```
$ umask -S  u=rwx,g=rx,o=
```

When you use the binary format, the mask is the inverse of the permissions you want to set. To set both the read and execute permission on and the write permission off, you use the octal number 2, a binary 010. To set all permissions on, you use an octal 0, a binary 000. The following example shows the mask for the permission defaults rwx, rx, and rx (rw, r, and r for files):

```
$ umask
0022
```

To set the default to only deny all permissions for others, you use 0027, using the binary mask 0111 for the other permissions.

```
$ umask 0027
```

Access Control Lists: ACL

Users can provide more refined control of their files and directories by using Access Control Lists (ACL). Access Control Lists maintains lists of users and groups and the rights they have to access certain files and directories. ACLs allow for much more refined access to directories, instead of just the all or nothing approach of owners, groups, and other. With ACLs only specific users could be granted write access to a file, while some others could be given just read access. Instead of opening a file up to all members of a group or everyone else on the system or network, an ACL could limit access to just a few specified users, regardless of their group membership. Like permissions, ACLs are controlled by the user, allowing users to setup access to a file by particular individuals or groups.

On Fedora, ACL support can be installed with the acl package. Check the Man pages for **acl**, **setfacl**, and **getfacl** for detailed descriptions and examples.

ACL entries have the three fields: a type (user, group, other, mask), a qualifier (specific user or group), and the discretionary access permissions (read, write, execute). The fields are separated by colons. The type can be u, **g**, **o**, and **m** referencing user, group, other, and mask, respectively. For the qualifier you can enter a user or group. Instead of a name you can use a user or group ID, UID or GID. Permissions can be read, write, or execute: **r**, **w**, or **x**. The permissions can be managed with binary or symbolic methods. In a standard binary method, the **rwx** permissions are positional, with read being first, write second, and execute third. If a dash (-) appears in any of these positions, it is denying that kind of access. **r-x** would allow read and execute permission, but deny write access. With a symbolic method, you use a + or ^ symbol to specify a permission you want to add or remove. **+r^w+x** would add read and execute permissions, and remove write permission.

The entry **u:chris:rw** would allow the user chris to have read and write permissions for the specified file. You can list several entries on the same line, separated by commas, or place each entry in a file on its own line.

ACL for file systems

File systems need to be mounted with the **acl** option. The ACL tools (acl package) include **setfacl** and **getfacl** commands to set permissions. See **setfacl** and **getfacl** Man pages for more information.

Once permissions are set for a file system, you can use the **mount** command with the **acl** option to mount it. With Fedora 10, be sure the file system is labeled.

```
sudo mount  -o acl  LABEL=myvideos /myvideo1
```

To have the device automatically mounted with the ACL options, add the acl option to its entry in the **/etc/fstab** file.

```
LABEL=myvideos     /myvideo1    ext3       defaults,acl    0     0
```

Displaying ACL controls

Once the file system has been mounted with ACL attributes enabled, you can then use the **getfacl** command to display the ACL attributes for particular files and directories. The **getfacl** command will list the file name, owner, and group. It will then list permissions for user, group, mask, and other. The permission entry will have three fields, the first being the type (user, group, mask, or other). The second is a list of users and groups that are permitted access. And the third is the permission granted to the listed users and groups. You can have several permissions of the same type, listing users and groups that would have different permissions. Default permissions are initially listed with empty user and group lists (empty second field).

```
getfacl  myfirstvid
```

To see the ACL for the current working directory, use the period as the argument.

```
getfacl .
```

When you use the **ls** command with the **–l** option on and ACL file or directory, a **+** will appear at the end of the permissions field indicating that ACLs are in effect.

```
ls -l myfirstvid
```

To copy an ACL file you use the **–p** permission (the move/rename operation, **mv**, will preserve the ACL permissions).

```
cp -p myfirstvid  myvidback
```

ACL settings

With the **setfacl** command you can control access by specific users, setting read, write, and execute permissions. Use the **-m** option to add new permissions and change current ones (**-x** removes a setting). As noted previously, users and permissions are referenced with a colon separate list with permissions specified using the **r**, **w**, and x options, and **u**, **g**, and **o** referencing user, group, and other categories. The argument **u:chris:rw** would allow the user chris to have read and write permissions for the specified file

```
setfacl  -m u:chris:rw  myfirstvid
```

The **getfacl** operation will then display added permission entry.

```
getfacl  myfirstvid
```

Tip: You can install and use the **eiciel** tool to provide a graphical user interface for setting ACL controls on files and directories.

To change a particular entry, use the **–m** option.

```
setfacl  -m u:chris:r  myfirstvid
```

To remove an entire entry, use the –x option. Specify just the type and the user or group name.

```
setfacl  -x u:chris  myfirstvid
```

Tip: You can also use chacl command to change ACL settings.

Instead of repeating the same ACL entries for each file, you can create a file that holds the entries and then read them to your file.

For each file ACL an effective mask is set up whenever you add a new entry. The effective rights mask is calculated by the **setfacl** command as the union of all other permissions already in specified for ACL entries, and the current one specified. In effect, when you use the **setfacl** command with the **-m** option to add an ACL entry, a mask entry is calculated by **setfacl** and added to the operation. You can turn off this implied calculation with the **-n** option. You would then have to explicitly set the effective rights mask.

```
setfacl -n –m u:chris:r,m::rw  myfirstvid
```

Should you use **chmod** to changes permissions on an ACL file, you are changing the mask entry. The mask entry takes precedence over all other entries. If you change a mask to just read only for all permissions, all other entries are overridden. This is indicated by an effective comment displayed after the entry, showing the actual permission allowed. The original entries remain, they are just not effective. There has been no modification of the original entry. This feature lets you shut off access to all users and groups, without having to change all the particular entries. You can then use **chmod** or change the mask entry directly to turn access back on for all the original entries.

Encrypted File Systems

Fedora lets you encrypt nonroot and swap file systems, allowing access only to those users with the appropriate encrypted password. You can apply encryption to both fixed and removable file systems such as USB devices. It is recommended that you use the LUKS (Linux Unified Key Setup) encryption tools to encrypt file systems. Fedora supports encrypted file system during installation.

To set up an encrypted file system is to use **cryptsetup** tool. You first use the **cryptsetup** command with the **luksFormat** option to initialize and create an encrypted volume. You will be prompted to specify a key (or add the key file as an argument). You will be prompted for the passphrase (or use **--keyfile** to specify the file with the passphrase). You can then format the file system, specifying its name and type.

Be sure the file system is not mounted. The default system type is **vfat**. Use the **-t** option to specify one. Be sure to specify the file system, the encryption cipher and passphrase, and the file system type and name. The **crypsetup** tool sets up the encrypted file system and you can use it directly, later formatting it with **mkfs**. Encryption is often used for removable devices like USB drives.

Once your encrypted file system is setup and formatted, restart your system. You can then access the encrypted partition or removable drive. For a USB drive or disk, from the file system window double-click the USB drive icon. This opens a window in which you are prompted for a password with the option to forget, remember for the session, or always remember. A message tells you the device is encrypted. Once you enter your password, you can then mount and access the device (double-click it again). The volume name will appear with an icon on your desktop. HAL will handle all mounting and access for removable media. Use the same procedure for fixed partitions. Instead of restarting your system after the initialization and format, you can use **crypsetup** with the **luksOpen** option to open the encrypted file system.

HAL manages access to encrypted file systems with the **15-lstorage-uks.fdi** file in the /**usr/share/hal/fdi/policy/10osverndor** directory. This file uses the **hal-luks-setup** and the **hal-luks-teardown** scripts to manage encrypted file systems.

If you want to manage fixed drives manually, you can place entries in the **/etc/crypttab** and **/etc/fstab** files for them.

The **cryptsetup** tool is installed as part of the **cryptsetup** package. Check **/usr/share/doc/cryptsetup** for HOWTOs and example for managing encrypted partitions. Support for encrypted file systems is provided by the **/etc/init.d/cryptdisks** script. Default options are set in the **/etc/defaults/cryptdisks** file which enable encrypted disk support.

If you did not use Luks, you will have to specify a encryption method with the **cipher** option. Use the **--cipher** option with **cryptsetup** in the **/etc/crypttab** entry. For an Encrypted Salt-Sector Initialization Vector (ESSIV) cipher, you use **aes-cbc-essiv:sha256**. For a plain cipher, you use **aes-cbc-plain**.

Security Audit Tool: sectool

When an attacker breaks into a system, they will usually try to gain control by making their own changes to system administration files, such as password files. They can create their own

user and password information, allowing them access at any time, or simply change the root user password. They can also replace entire programs, such as the login program, with their own version. One method of detecting such actions is to use an audit and integrity checking tool such as the Fedora Security Audit Tool (sectool), You can find out more about sectool, including documentation and examples, at **https://fedorahosted.org/sectool/**

Note:Tripwire and AIDE also detect any changes to system administration files. AIDE (Advanced Intrusion Detection Environment) is a free and enhanced alternative to Tripwire. Both are available on the Fedora repository.

The Security Audit Tool can perform numerous tests on your system, including integrity checks and specific file, directory, or service checks (see Figure 11-3). Checks can be performed for different runlevel configurations (1 through 5), with defaults already selected for you. Tests are listed on the left pane, beginning with the integrity check. On the left pane is an Overview tab describing the test, and a Test Results tab has three panes, the Test and its result, a Summary pane, and a lower pane listing the Severity and test, and description of problems found.

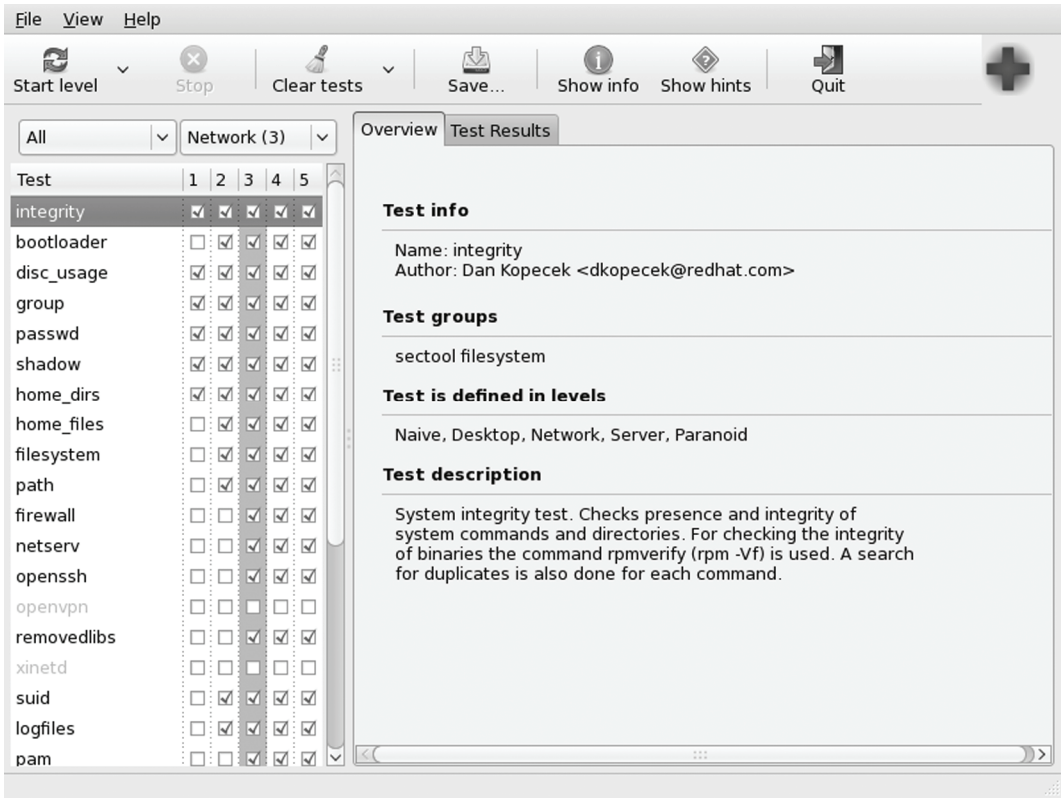


Figure 11-3: Sectool

To run a specific test, select it and then choose "Start | Start the selected test" from the File menu. You can also right-click and select "Start the selected test".

On the left pane listing the test, you can use the pop-up menu to select different groups of tests, such as file system, users, network or daemons. The sectool group performs just the integrity check.

A second pop-up menu lets you set the degree of testing. You can choose from Naive, Desktop, Network, Server, and Paranoid. What may be considered a warning in one may be passed in another.

An integrity checking tool works by first creating a database of unique identifiers for each file or program to be checked. These can include features such as permissions and file size, but more importantly, they can also include checksum numbers generated by encryption algorithms from the file's contents. Default identifiers are checksum numbers created by algorithms such as the SHA2 modification digest algorithm. An encrypted value that provides such a unique identification of a file is known as a signature. In effect, a signature provides an accurate snapshot of the contents of a file. Files and programs are then periodically checked by generating their identifiers again and matching them with those in the database. The intrusion detection application will generate signatures of the current files and programs and match them against the values previously generated for its database. Any differences are noted as changes to the file, and you are notified of the changes.

Note: You can also check your log files for any suspicious activity. The **/var/log/messages** file in particular is helpful for checking for critical events such as user logins, FTP connections, and superuser logins.

fedora

12. Security Enhanced Linux

Although numerous security tools exist for protecting specific services, as well as user information and data, no tool has been available for protecting the entire system as the administrative level. Security-Enhanced Linux is a project to provide built-in administrative protection for aspects of your Linux system. Instead of relying on users to protect their files or on a specific network program to control access, security measures would be built into the basic file management system and the network access methods. All controls can be managed directly by an administrator as part of Linux system administration.

Security-Enhanced Linux (SELinux) is a project developed and maintained by the National Security Agency (NSA), which chose Linux as its platform for implementing a secure operating system. Most Linux distributions have embraced SELinux and have incorporated it as a standard feature. Detailed documentation available from resources listed in Table 12-1, including sites provided by the NSA and SourceForge. Also check the Fedora Project sites for manuals, FAQs, or documentation on SELinux**:** **http://fedoraproject.org/wiki/SELinux** and **http://docs.fedoraproject.org/selinux-user-guide/**.

Resource	Location
Fedora Project SELinux FAQ	**http://fedoraproject.org/wiki/SELinux**
Fedora 10 SELinux User Guide	**http://docs.fedoraproject.org/selinux-user-guide/**
NSA SELinux	**http://www.nsa.gov/selinux**
NSA SELinux FAQ	**http://www.nsa.gov/selinux/info/faq.cfm**
SELinux at sourceforge.net	**http://selinux.sourceforge.net**
Writing SELinux Policy HOWTO	Accessible from "SELinux resources at sourceforge" link at **http://selinux.sourceforge.net**
NSA SELinux Documentation	**http://www.nsa.gov/selinux/info/docs.cfm**
Configuring SELinux Policy	Accessible from NSA SELinux Documentation
SE Linux Reference Policy Project	**http://oss.tresys.com/projects/refpolicy**

Table 12-1: SELinux Resources

Linux and Unix systems normally use a discretionary access control (DAC) method for restricting access. In this approach users and the objects they own, such as files, determine permissions. The user has completed discretion over the objects it owns. The weak point in many Linux/Unix systems has been the user administrative accounts. If an attacker managed to gain access to an administrative account, they would have complete control over the service the account managed. Access to the root user would give control over the entire system, all its users, and any network services it was running. To counter this weakness, the NSA set up a mandatory access control (MAC) structure. Instead of an all-or-nothing set of privileges based on accounts, services and administrative tasks are compartmentalized and separately controlled with policies detailing what can and cannot be done. Access is granted not just because one is an authenticated user, but when specific security criteria are met. Users, applications, processes, files, and devices can be given just the access they need to do their job, and nothing more.

Flask Architecture

The Flask (Flux Advanced Security Kernel) architecture organizes operating system components and data into subjects and objects. Subjects are processes: applications, drivers, system tasks that are currently running. Objects are fixed components such as files, directories, sockets, network interfaces, and devices. For each subject and object, a security context is defined. A *security context* is a set of security attributes that determine how a subject or object can be used. This approach provides a very fine-grained control over every element in the operating system as well as all data on your computer.

The attributes designated for the security contexts and the degree to which they are enforced are determined by an overall security policy. The policies are enforced by a security server. Distributions may provide different pre-configured policies from which to work. For example, Fedora provides three policies each in its own package, strict, targeted and mls, which are all variation of a single reference policy.

SELinux use a combination of the Type Enforcement (TE), Role Based Access Control (RBAC), and Multi-Level Security (MLS) security models. Type Enforcement focuses on objects and processes like directories and applications, whereas Role Based Access Enforcement controls user access. For the Type Enforcement model, the security attributes assigned to an object are known as either domains or types. Types are used for fixed objects such as files, and domains are used for processes such as running applications. For user access to processes and objects, SELinux make use of the Role Based Access Control model. When new processes or objects are created, transition rules specify the type or domain they belong to in their security contexts.

With the RBAC model, users are assigned roles for which permissions are defined. The roles restrict what objects and processes a user can access. The security context for processes will include a role attribute, controlling what objects it can assess. The new Multi-Level Security (MLS) adds security level, containing both a sensitivity and capability value.

Users are given separate SELinux user identities. Normally these correspond to the user IDs set up under the standard Linux user creation operations. Though they may have the same name, they are not the same identifiers. Standard Linux identities can be easily changed with commands like **`setuid`** and **`su`**. Changes to the Linux user ID will not affect the SELinux ID. This means that even if a user changes its ID, SELinux will still be able to track it, maintaining control over that user.

SE Linux Policy Packages

Fedora Linux provides several SELinux policy packages. The targeted one is installed by default. You can use Pirut to download the others. The source code, along with the source code documentation, is now kept in separate RPMS packages, which you download and manually install.

Note: You can also use **setroubleshoot** to check and locate problems you may be having with SELinux.

Each policy installs its configuration files in **/etc/selinux**. If you want to add your own module, you need to use the policy headers and module .pp files, which are located in **/usr/share/selinux**. The **selinux-development** package installs the main development module headers in **/usr/share/selinux/devel**. Other packages, such as targeted-policy, then install their own module **.pp** files, with links to the **devel** directory.

selinux-policy	SELinux documentation and support, man pages and /usr/share/selinux
selinux-policy-targeted	SELinux targeted policy configuration
selinux-policy-strict	SELinux strict policy configuration
selinux-policy-mls	SELinux MLS policy configuration
selinux-policy-*version*-src.rpms	SELinux Reference Policy source files
selinux-doc	Binary documentation
selinux-doc-*version*-src.rpms	SELinux Reference Policy documentation
selinux-policy-devel	SELinux development modules, **/usr/share/selinux/devel**

Check the **/usr/share/doc/selinux*** directory for examples on how to create a module of your own.

Keep in mind that these are not the source files. There are no **.te** files. The source files have to be installed separately from the **selinux-policy-*version*-src.rpms** file located in the **source/SRPMS** directory of the Fedora distribution. You will have to manually access the site and download them directly using a Web browser like Firefox or an FTP client. The source packages will not show up on Yum.

So there are two selinux-policy creation packages, which are very different from each other, one RPM and one source: **selinux-development** and **selinux-policy-*version*-src.rpms**. The RPM one, which you can install with Pirut or Yum, contains just the headers and module **.pp** files needed so that you can compile your own additional modules. The selinux-policy source SRPMS file is the entire SELinux reference policy with all the source code files, including **.te** files for each module.

You see the same kind of structure for kernel modules. The selinux-development RPM contains only the **header/pp** files needed to create modules, much as you need only the kernel headers to create kernel modules, not the entire kernel source.

System Administration Access

It is critically important that you make sure you have system administrative access under SELinux before you enforce its policies. This is especially true if you are using a strict or mls policy, which imposes restrictions on administrative access. You should always use SELinux in permissive mode first and check for any messages denying access. With SELinux enforced, it may no longer matter whether you can access the root user or not. What matters is whether your user, even the root user, has sysadm_r role and sysadm_t object access and an administrative security level. You may not be able to just use the **su** command to access the root user and expect to have root user administrative access. Recall that SELinux keeps its own security identities that are not the same as Linux user IDs. Though you might change your user ID with **su**, you still have not changed your security ID.

The targeted policy will set up rules that allow standard system administrator access using normal Linux procedures. The root user will be able to access the root user account normally. In the strict policy, however, the root user needs to access its account using the appropriate security ID. Both are now part of a single reference policy. If you want administrative access through the **su** command (from another user), you would first use the **su** command to log in as the root user. You then have to change your role to that of the sysadm_r role. A user can have several allowed possible roles it could assume.

To change the role, you use the **newrole** command with the **-r** option.

```
newrole -r sysadm_r
```

Terminology

SELinux uses several terms that have different meanings in other contexts. The terminology can be confusing because the some of the terms, such as domain, have different meanings in other, related, areas. For example, a domain in SELinux is a process as opposed to an object, whereas in networking the term refers to network DNS addresses.

Identity

SELinux creates identities with which to control access. Identities are not the same as traditional user IDs. At the same time, each user normally has an SELinux identity, though the two are not linked. Affecting a user does not affect the corresponding SELinux identity. SELinux can set up a separate corresponding identity for each user, though on the less secure policies, such as targeted policies, general identities are used. A general user identity is used for all normal users, restricting users to user-level access, whereas administrators are given administrative identities. You can further define security identities for particular users.

The identity makes up part of a security context that determines what a user can or cannot do. Should a user change user IDs, that user's security identity will not change. A user will always have the same security identity. In traditional Linux systems, a user could use commands like **su** to change their user ID, becoming a different user. On SELinux, even though a user could still change his or her Linux user ID, the user still retains the same original security ID. You always know what a particular person is doing on your system, no matter what user ID that person may assume.

The security identity can have limited access. So even though a user may use the Linux **su** command to become the root user, the user's security identity could prevent his or her from performing any root user administrative commands. As noted previously, to gain an administrative access, the role for their security identity would have to change as well.

Use **id -Z** to see what the security context for your security identity is, what roles you have, and what kind of object you can access. This will list the user security context that starts with the security ID, followed by a colon, and then the roles a user has and the objects the user can control. Security identities can have roles that control what they can do. A user role is user_r, and a system administration role is system_r. The general security identity is user_u, whereas a particular security identity will normally use the user name. The following example shows a standard user with the general security identity:

```
$ id -Z
 user_u:user_r:user_t
```

In this example the user has a security identity called george:

```
$ id -Z
 george:user_r:user_t
```

You can use the **newrole** command to change the role a user is allowed. Changing to a system administrative role, the user can then have equivalent root access.

```
$ newrole -r sysadm_r
$ id -Z
 george:sysadm_r:sysadm_t
```

Domains

Domains are used to identify and control processes. Each process is assigned a domain within which it can run. A domain sets restrictions on what a process can do. Traditionally, a process was given a user ID to determine what it could do, and many had to have root user ID to gain access to the full file system. This also could be used to gain full administrative access over the entire system. A domain, on the other hand, can be tailored to access some areas but not others. Attempts to break into another domain, say the administrative domain, would be blocked. For example, the administrative domain is sysadm_t, whereas the DNS server uses only named_t and users have a user_t domain.

Types

Whereas domains control processes, *types* control objects like files and directories. Files and directories are grouped into types that can be used to control who can have access to them. The type names have the same format as the domain names, ending with a _t suffix. Unlike domains, types reference objects, including files, devices, and network interfaces.

Roles

Types and domains are assigned to roles. Users (security identities) with a given role can access types and domains assigned to that role. For example, most users can access user_t type objects, but not sysadm_t objects. The types and domains a user can access are set by the role entry in configuration files. The following example allows users to access objects with the user password type:

```
role user_r types user_passwd_t
```

Security Context

Each object has a security context that set its security attributes. These include identity, role, domain or type. A file will have a security context listing the kind of identity that can assess it, the role under which it can be accessed, and the security type it belongs to. Each component adds its own refined level of security. Passive objects are usually assigned a generic role, **object_r**, which has no effect, as such objects cannot initiate actions.

For a normal file created by users in their own directories, you would have the following identity, role, and type. The identity is a user and the role is that of an object. The type is the user's home directory. This type is used for all subdirectories and their files created within a user's home directory.

```
user_u:object_r:user_home_t
```

A file or directory created by that same user in a different part of the file system will have a different type. For example, the type for files created in the **/tmp** directory will be tmp_t.

```
user_u:object_r:tmp_t
```

Transition: Labeling

A *transition,* also known as labeling, assigns a security context to a process or file. For a file, the security context is assigned when it is created, whereas for a process the security context is determined when the process is run.

Making sure every file has an appropriate security context is called *labeling.* Adding another file system would require that you label (add security contexts) to the directories and files on it. Labeling varies, depending on the policy you use. Each policy may have different security contexts for objects and processes. Relabeling is carried out using the **`fixfiles`** command in the policy source directory.

```
fixfiles relabel
```

Policies

A *policy* is a set of rules to determine the relationships between users, roles, and types or domains. These rules state what types a role can access and what roles a user can have.

Multi-layer Security (MLS) and Multi-category Security (MCS)

Multi-layer security (MLS) adds a more refined security access method, designed for servers. MLS adds a security level value to resources. Only users with access to certain levels can access the corresponding files and applications. Within each level access can be further controlled with the use of categories. Categories work much like groups, allowing access only to users cleared for that category. Access becomes more refined, instead of an all-or-nothing situation.

Multi-category security (MCS) extends SE Linux not only to use by administrators, but also by users. Users can set categories which can restrict and control access to their files and applications. Though based on MLS, it uses only categories, not security levels. Users can select a category for a file, only the administrator can create a category and determine what users can access it. Though similar in concept to an ACL (Access Control List), it differs in that it makes use of the SE Linux security structure, providing user-level control enforced by SE Linux.

Labeling

One of the most common administrative tasks for controlling access is relabeling a file's security context. A file with a certain context may deny access to an application that is set up to access files of a different kind of access. You use the chcon, semanage with the fcontext option, and restorecon to change a file's security context. The setfiles command is used for relabeling.

chcon Changes contexts, but does not persist if a relabel or restore operation is performed.

restorecon Restores the default security context of a file.

semanage fcontext Change the security context of a file. This change is persistent through any relabel or restore operations. Changes are added to the active policy's **file_contexts** configuration file, like **/etc/selinux/targeted/contexts/files/file_contexts** for the targeted policy.

For example, in the case of user Web site directories, the Web server, which is designed to access files with Web contexts, will try to access files that have a user context. The user Web site files are created in a user's home directory by the user and are thereby given the user_home_t context type. The Web server can only access files with the httpd_sys_content_t type.

Using the chcon command would change the type to **httpd_sys_content_t**, but any restore or relabel operations would change it back to the default **user_home_t** type. Use the **-t** option to indicate the file context type. You can add the **-R** option for changing any files and subdirectories already in the directory.

```
chcon -Rt httpd_sys_content_t public_html
```

To make the change persistent you have to add an entry for in the policy's **file_contexts** file. Use **semanage** with the **fcontext** directive and the **-a** (add) and **-t** (type) options. The file_contexts configuration file is read by the restorecon command. Once you add a new entry, run restorecon to make the actual change. Be sure to use full pathnames for file and directory names.

```
semanage  fcontext -a -t httpd_sys_content_t /home/robert/public_html
restorecon  /home/robert/public_html
```

Any later relabeling or restore operations will use that new file context configuration for this file.

Also, any files or directories you create will inherit the security context of the parent directory. In this example, any files and subdirectories created later in the **public_html** directory will also have the context type **httpd_sys_content_t**. Any previous files or directories will not.

TIP: The tar command does not retain security context when creating an archive of files and directories. Use the **tar** command with the **--selinux** option to retain the file contexts for files in the archive in the archive.

To change any files and subdirectories already in the directory, you use file matching regular expressions, **(/.*)**, for the fcontext configuration. Be sure to quote the entire name including the expression. When you run the **restorecon** command to make the change add the **-R** option (recursive) to change any files and subdirectories also. Changes are made to the directory and to any of its files and subdirectories.

```
semanage  fcontext -a -t "httpd_sys_content_t /home/robert/public_html(/.*)"
restorecon  -v -R /home/robert/public_html
```

To return to the default context, you would first remove the configuration entry, and then run restorecon. Use **semanage fcontext** with the **-d** option to delete a file's context. If the entry uses regular expressions, like (/.*), be sure to quote the entire file or directory name, including the expression.

```
semanage fcontext -d "/home/robert/public_html(/.*)"
restorecon -R /home/robert/public_html
```

NFS, Samba, and mount

A similar issue occurs commonly with NFS mounted file systems. NFS file systems use the nfs_t file context by default. If you want to make that file systems part of a Web server's Web site files, you would have to change the type to **httpd_sys_content_t**. NFS though is a remote system, and on its remote system will have its own file contexts. You would want to change the context only temporarily when the file system is mounted by your system. To do this you would use the context option for the **mount** command.

```
mount mynfs:/myrw/  /myweb/myw/  -o  context="system_u:object_r:httpd_sys_context_t:s0"
```

To make the operation automatic, add an entry for it in the **/etc/fstab** file.

```
mynfs:/myrw/  /myweb/myw/  nfs  context="system_u:object_r:httpd_sys_context_t:s0" 0  0
```

In the case of Samba, problems can occur when you create a new file or directory on a mounted Samba share. By default the new file would have the context type **default_t**. The Samba server though expects a context type of **samba_share_t**. When a Samba share is mounted, all its files and directories are given the **samba_share_t** context type. To overcome this problem, you can set the default context type when the file system is mounted. Then, any new files will have this context type. For Samba shares you can set the default context type to **samba_share_t**. Use the defcontext type with the mount option to se the default context type.

```
mount /dev/sd4 /mywin1 -o defcontext="system_u:object_r:samba_share_t:s0"
```

To make the operation automatic, add an entry for it in the **/etc/fstab** file.

```
/dev/sd4 /mywin1 ntfs context="system_u:object_r:samba_share_t:s0" 0 0
```

Copy and Moving Files: cp and mv

When copying or moving files you can choose to change or not change the original file's contexts.

By default the cp command will use the parent directory's contexts for the new created copies, not the contexts from the original copies. This is in keeping with the policy of newly created files and directories inheriting their contexts from their parent directory. If you are overwriting files, then the context of the overwritten file is used, not the original files.

If you want to retain the original file's contexts in the copies, use the **--preserve=context** option.

```
cp --preserve=context
```

If you want to change the copies to a particular file context, you can use the -Z option.

```
cp -Z system_u:object_r:httpd_sys_context:s0 /mytest/test1  /myweb/test.html
```

A move operation, **mv**, will retain the original context. The move operation is literally just a rename. The physical files are the same.

Management Operations for SELinux

Certain basic operations such as checking the SELinux status, checking a user's or file's security context, or disabling SELinux at boot can be very useful.

Turning off SELinux

Should you want to turn off SELinux before you even start up your system, you can turn it off at the boot prompt. Just add the following parameter to the end of your GRUB boot line.

```
selinux=0
```

To turn off SELinux permanently, you can use the system-config-selinux Status tab to set the System Default Enforcing Mode to Disabled, or set the **SELINUX** variable in the **/etc/selinux/config** file to **disabled**.

```
SELINUX=disabled
```

To turn off (permissive mode) SELinux temporarily without rebooting, use the **setenforce** command with the 0 option; use 1 to turn it back on (enforcing mode). You can also use the terms permissive or enforcing at the arguments instead of 0 or 1. You must first have the sysadm_r role, which you can obtain by logging in as the root user.

```
setenforce 1
```

Checking Status and Statistics

To check the current status of your SELinux system, you can use **sestatus** . Add the **-v** option will also display process and file contexts, as listed in /etc/sestatus.conf. The contexts will specify the roles and types assigned to a particular process, file, or directory.

```
sestatus -v
```

Use the **seinfo** command to display your current SELinux statistics.

```
# seinfo
Statistics for policy file: /etc/selinux/targeted/policy/policy.21
Policy Version & Type: v.21 (binary, MLS)

   Classes:            55    Permissions:       206
   Types:            1043    Attributes:         85
   Users:               3    Roles:               6
   Booleans:          135    Cond. Expr.:       138
   Sensitivities:       1    Categories:        256
   Allow:           46050    Neverallow:          0
   Auditallow:         97    Dontaudit:        3465
   Role allow:          5    Role trans:          0
   Type_trans:        987    Type_change:        14
   Type_member:         0    Range_trans:        10
   Nodecon:             8    Initial SIDs:        0
```

Checking Security Context

The **-Z** option used with the **ls**, **id**, and **ps** commands can be used to check the security context for files, users, and processes respectively. The security context tells you the roles that users must have to access given processes or objects.

```
ls -lZ
id -Z
ps -eZ
```

SELinux Management Tools

SELinux provides a number of tools to let you manage your SELinux configuration and policy implementation, including semanage to configure your policy. The **setools** collection provides SELinux configuration and analysis tools including apol, the Security Policy Analysis tool, for domain transition analysis, sediffx for policy differences, and **seaudit** to examine the auditd logs. (see Table 12-2). The command line user management tools, useradd, usermod, and userdel, all have SELinux options that can be applied when SELinux is installed. In addition, the **audit2allow** tool will convert SELinux denial messages into policy modules that will allow access. See a complete listing of SELinux commands at
www.fedoraproject.org/wiki/SELinux/Commands

Command	Description
`seinfo`	Display policy statistics
`sestatus`	Check status of SELinux on your system, including the contexts of processes and files
`sesearch`	Search for type enforcement rules in policies
`seaudit`	Examine SELinux log files
`sediffx`	Examine SELinux policy differences
`setroubleshoot`	SELinux GUI troubleshooting tool
`system-config-selinux`	Fedora SELinux GUI configuration tool
`autid2allow`	Generate policy allow rules for modules using audit AVC denial messages.
`apol`	SELinux Policy Analysis
`checkpolicy`	The SELinux policy compiler
`fixfiles`	Check file systems and set security contexts
`restorecon`	Set security features for particular files
`newrole`	New role
`setfiles`	Set security context for files
`chcon`	Change context
`chsid`	Change security ID

Table 12-2: SELinux Tools

With the modular version of SELinux, policy management is no longer handled by editing configuration files directly. Instead you use the SELinux management tools such as the command line tool semanage and system-config-selinux (System | Administration | SELinux Management). These tools make use of interface files to generate changed policies.

Configuration with system-config-selinux

With system-config-selinux you can manage and configure your SELinux policies, though you cannot create new policies (see Figure 12-1). You can access system-config-selinux from the System | Administration menu by selecting the SELinux Management entry.

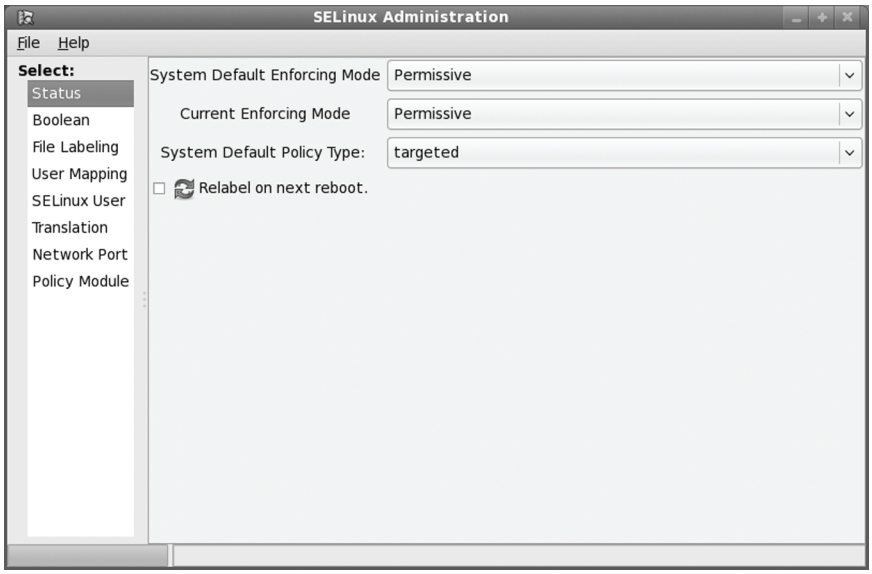


Figure 12-1: The system-config-selinux: Status

The system-config-selinux window will list several panes with a sidebar menu for Status, Boolean, File Labeling, User Mapping, SELinux User, Translation, Network Port, and Policy Module. system-config-selinux will invoke the SELinux management tools like **sestatus** and **semanage** with appropriate options to make configuration changes.

The Status pane lists pop-up menus for selecting policy and enforcement defaults, as well as the current enforcing mode. A check box to relabel on reboot lets you force relabeling of your file security contexts when you reboot (see Figure 12-1).

The Boolean pane lists various options for targeted services like Web and FTP servers, NFS, and Samba (see Figure 12-2). With these you can further modify how each service is controlled. There are expandable menus for different services like FTP, Apache Web server, and Samba. For example, the FTP entries lets you choose whether to allow access to home directories or to allow NFS file transfers. The number of boolean entries is extensive. To see just the ones you have activated, you can click the Customized button. Only selected boolean options will be displayed. When you click the Customized button, it changes to the All button and the Revert button becomes active. Revert will restore system boolean defaults. The Lockdown button will start the Lockdown wizard, opening a new window with Booleans displayed in an expandable tree on a left pane, and the right pane gives you a choice of Enable, Disable, and Default option for the selected boolean. Forward and Backward buttons moves you sequentially through the list of Booleans. The Lockdown wizard will lockdown an SELinux system, preventing any modification

to the boolean configuration. From the File menu you can select Apply to directly apply the configuration, or save the configuration as a different file.

The File Labeling pane will list your system directories and files, showing their security contexts and types. You can edit a file's properties by selecting the entry and then clicking Properties. This displays a dialog with the File Name, Type, SELinux Type, and MLS Level. You can change the SELinux type or the MLS level. For a permissive policy, the MLS level will be s0, allowing access to anyone. You can also add or delete entries.

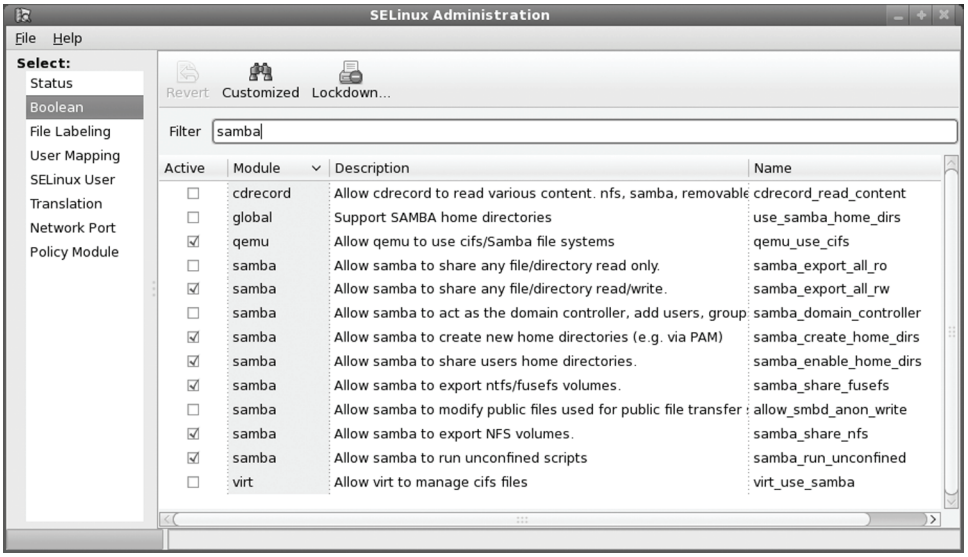


Figure 12-2: The system-config-selinux: Boolean pane

User Mapping shows the mapping of user login names to SELinux users. Initially there will be two mappings: the root user and the default user.

The SELinux Users pane shows the different kinds of SELinux users. Initially there will be three user types: root, system_u, and user_u (see Figure 12-3). The root user is the root user, which has full and total administrative access to the entire system. The system_u user allows users to take on administrative access where needed. The user_u user is used for normal users. Each entry lists its SELinux User, SELinux prefix, MLS level, MLS range, and SELinux roles. MLS level is the access level (s0 on a permissive policy), and MLS range is the range of access from SystemLow to SystemHigh. A given user has certain roles available. The root user has the system_r, sysadm_r, and user_r roles, allowing it system access, administration capability, and standard user access. The same roles are applied to user_u, allowing the user to perform system administration if that user has the root user password. The system_u role has only system access and cannot perform system administration.

The Translation pane lets you set MLS symbols. Initially you will have symbols for SystemHigh and the SystemLow–SystemHigh range. You change the MLS levels for a mapping, changing security level access across the system.

The Network Port pane lists the network protocol, the SELinux type, and the MLS security level for ports on your system. Select an entry and click Properties to change the SELinux type or the MLS level for the port. The Group View button will display the SELinux type along with a list of the ports they apply to. This view does not display the MLS level, as these apply to ports individually.

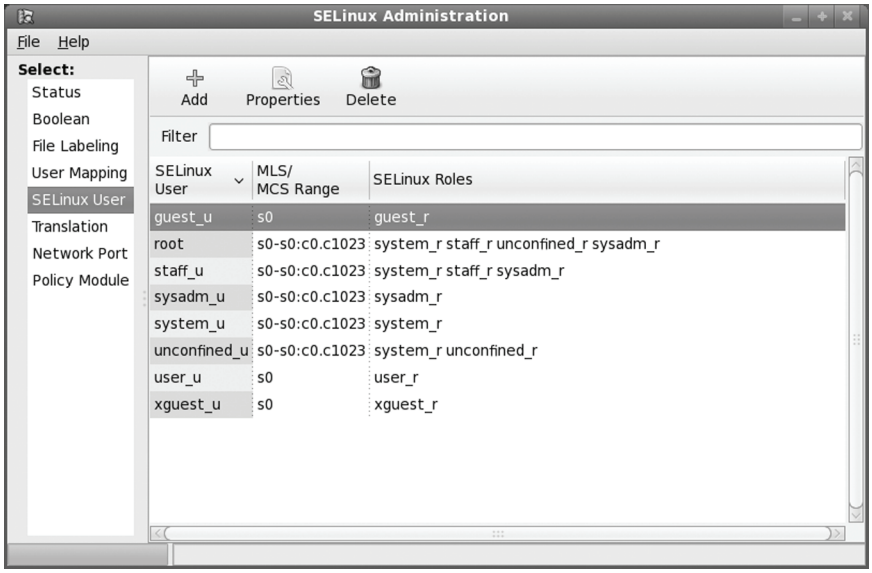


Figure 12-3: system-config-selinux SELinux User pane

The Policy modules pane lists the different SELinux policy modules. Here you will see modules for different applications like Thunderbird and Evolution, as well as device service like USB and HAL. Listed also are desktops like GNOME. The pane allows you to add or remove a module. You can also enable or provide additional audit rules for a module for logging.

SELinux Troubleshooting and audit2allow

Fedora includes the SELinux troubleshooter which notifies problems that SELinux detects. Whenever SELinux denies access to a file or application, the kernel issue an AVC notice. These are analyzed by SELinux troubleshooter to detect problems that users may have to deal with. When a problem is detected, the SELinux troubleshooter notification will be displayed in the desktop notification area along with the troubleshooter icon. Clicking on the icon or notice will open the SELinux troubleshooter window. You can also access it at any time from System | Administration | SELinux troubleshooter. You can find out more information about SELinux troubleshooter at **http://hosted.fedoraproject.org/projects/setroubleshoot**.

The SELinux troubleshooter window will display a list of notices, along with their date, the number of times it has occurred, its category, and brief explanation. The Filter entry lets you turn off future notification of this event. Selecting and entry will display detailed information about

the notice in four sections: Summary, Detailed Description, Allowing Access, and Additional Information (see Figure 12-4)

In many cases the problem may be simple to fix, as shown in the Allowing Access section in Figure 12-4. Often, the security context of a file has to be renamed to allow access. You use the **chcon** command to change a file's security context. In this rename access needs to be granted to the Samba server for a **log.richard3** file in the **/var/lib/samba** directory. The SELinux troubleshooter will take no action of its own. Instead it recommends possible actions. In this example, the user just issues the following **chcon** command.

```
chcon -R -t samba_share_t log.richard3
```

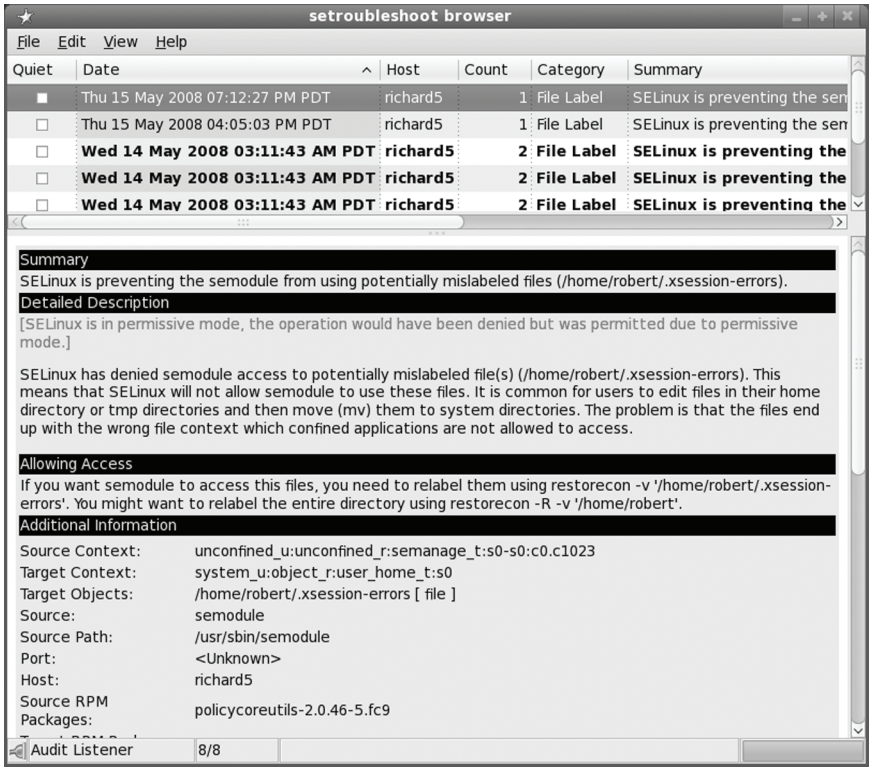


Figure 12-4: SELinux troubleshooter

More complicated problems, especially ones that are unknown, may require you to create a new policy module using the AVC messages in the audit log. To do this, you can use the **audit2allow** command. The command will take an audit AVC messages and generate commands to allow SELinux access. The audit log used on Fedora is **/var/log/audit/audit.log**. This log is output to **audit2allow** which then can use its **-M** option to create a policy module. I

```
cat /var/log/audit/audit.log | audit2allow -M local
```

You then use the **semodule** command to load the module.

```
semodule -i local.pp
```

If you wan to first edit the allowable entries, you can use the following to create a **.te** file of the local module, **local.te**, which you can then edit.

```
audit2allow -m local -i  /var/log/audit/audit.log   >  local.te
```

Once you have edited the **.te** file, you can then use **checkmodule** to compile the module, and then **semodule_package** to create the policy module, **local.pp**. Then you can install it with **semodule**. You first create a **.mod** file with **checkmodule**, and then **.pp** file with **semodule_package**.

```
checkmodule -M -m -o local.mod local.te
semodule_package -o local.pp  -m local.mod
semodule -i local.pp
```

In this example the policy module is just called **local**. If you later want to create a new module with **audit2allow**, you should either use a different name or just append the output to the **.te** file using the **-o** option.

SELinux Policy Generation Tool

The SELinux Policy Generation tool generates policy frameworks to allow SELinux to control specific applications or users. On the initial screen, you select the type of application, user, or the root user (system administrator), see Figure 12-5.

Figure 12-5: SELinux Policy Generation Tool: policy type

For an application you are then prompted to enter the application name, the location of its executable file, and, for a standard daemon (stand alone server), the init script it uses. For the executable file, you can click on a file manger button to the right of the text box to open a selection window for the **/usr/sbin** directory. For a standard daemon the selection window opens to the **/etc/rc.d/init.d** directory, listing all the daemon start up scripts.

On the following screens you enter the ports the application listens on and the ones it connects to. Then you select common traits like sending email or using PAM authentication. Then select the files and directories the application uses, along with the Booleans it may use. Choose

where to save the policy file. The rules generated will then be listed. When you click Apply, the policy files will be generated (**.te**, **.fc**, **.sh**, and .**if**).

After clicking Apply and saving your configuration files, you can click the Back button until you return to the first window. Then you can select another type of role for another policy.

For users, when you select Exiting User Roles, a list of exiting user roles will be displayed, like guest_u or user_u. On the next screen you can choose addition domains the user can access like Wine, polkit, or games. You can then select additional roles for the user, like sysadm and webadm. You then enter the ports the user can listen and connect to, as well as any Booleans that may apply. Choose where to save the policy file. If you have already selected one, it will be displayed. The rules generated will then be listed. When you click Apply, the policy files will be generated (**.te**, **.fc**, **.sh**, and .**if**).

For User Role, you will be prompted to enter a name for the role, then select the domains it can access, followed the ports used, and any booleans you want to apply. Upon clicking Apply, the policy files will be generated and saved.

For the Root Admin User Role, you first enter a name for the name for the user role, and then you select the domains this user role can administer, like bind, cups, and ntp. Use the Ctrl key to select more than one. Then select the user roles that can transition to this domain. Again select the ports listened and connected to, if any, along with any Boolean. Click Apply to generate the configuration files.

semanage

semanage lets you change your SELinux configuration without having to edit selinux source files directly. It covers several major categories including users, ports, file contexts and logins. Check the man page for semanage for detailed descriptions. Options let you modify specific security features such as **-s** for the user name, **-R** for the role, -t for the type, and -r for an MLS security range. The following example adds a user with role user_r.

```
semanage user -a -R user_r  justin
```

semange is configured with the **/etc/selinux/semanage.conf** file where you can configure semange to write directly on modules (the default) or work on the source.

apol, The Security Policy Analysis Tool

The SE Linux Policy Analysis tool, **apol**, provides a complex and detailed analysis of a selected policy (**setools** package). Select the SELinux Policy Analysis entry in the Applications | System Tools menu to start it. You will also have to locate and open the SELinux policy file in **/etc/selinux/** directory. You could also use the SELinux Policy Difference tool to compare policies, and the SELinux Audit Log Analysis tool for log analysis. These are third party tools, though, and not directly supported by Fedora.

Checking SELinux Messages: seaudit

SELinux AVC messages are now saved in the **/var/log/audit/audit.log** file. These are particularly important if you are using the permissive mode to test a policy you want to later enforce. You need to find out if you are being denied access where appropriate and afforded control when needed. To see just the SELinux messages, you can use the seaudit tool (System |

Administration | SELinux Audit Log Analysis). Start up messages for the SE Linux service are still logged in **/var/log/messages**.

Allowing access: chcon and audit2allow

Whenever SELinux denies access to a file or application, the kernel issue an AVC notice. In many cases the problem can be fixed by just renaming the security context of a file to allow access. You use the **chcon** command to change a file's security context. Access needs to be granted to the Samba server for a **log.richard3** file in the **/var/lib/samba** directory. The SELinux troubleshooter will take no action of its own. Instead it recommends possible actions. In this example, the user just issues the following **chcon** command.

```
chcon -R -t samba_share_t log.richard3
```

More complicated problems, especially ones that are unknown, may require you to create a new policy module using the AVC messages in the audit log. To do this, you can use the **audit2allow** command. The command will take an audit AVC messages and generate commands to allow SELinux access. The audit log is **/var/log/audit/audit.log**. This log is output to **audit2allow** which then can use its **-M** option to create a policy module. I

```
cat /var/log/audit/audit.log | audit2allow -M local
```

You then use the semodule command to load the module.

```
semodule -i local.pp
```

If you wan to first edit the allowable entries, you can use the following to create a **.te** file of the local module, **local.te**, which you can then edit.

```
audit2allow -m local -i  /var/log/audit/audit.log   >  local.te
```

Once you have edited the **.te** file, you can then use **checkmodule** to compile the module, and then **semodule_package** to create the policy module, **local.pp**. Then you can install it with **semodule**. You first create a **.mod** file with **checkmodule**, and then **.pp** file with **semodule_package**.

```
checkmodule -M -m -o local.mod local.te
semodule_package -o local.pp  -m local.mod
semodule -i local.pp
```

In this example the policy module is just called **local**. If you later want to create a new module with **audit2allow**, you should either use a different name or just append the output to the **.te** file using the **-o** option.

Tip: On Red Hat and Fedora distributions you can use the SELinux troubleshooter (setroubleshooter) to detect SELinux access problems.

The SE Linux reference policy

A system is secured using a policy. SE Linux now uses a single policy, the reference policy, instead of the two separate targeted and strict policies used in previous editions (see **http://oss.tresys.com/projects/refpolicy**). Instead of giving users just two alternatives, strict and targeted, the SE Linux reference policy project aims to provide a basic policy that can be easily adapted and expanded as needed. The SE Linux reference policy configures SE Linux into modules

that can be handled separately. You still have strict and targeted policies, but these are variations on a basic reference policy. In addition you can have an MLS policy for multi layer security. The targeted policy is installed by default, and you can install the strict or MLS policies yourself.

On some distributions, like Fedora, there may be separate policy configurations already provided. For example, Fedora currently provides three effective policies: **targeted**, **strict**, and **mls**. The targeted policy is used to control specific services, like network and Internet servers such as Web, DNS, and FTP servers. It also can control local services with network connections. The policy will not affect just the daemon itself, but all the resources it uses on your system.

The strict policy provides complete control over your system. It is under this kind of policy that you users and even administrators can be inadvertently locked out of the system. A strict policy needs to be carefully tested to make sure access is denied and granted where appropriate.

There will be **targeted**, **strict**, and **mls** subdirectories in your **/etc/selinux** directory, but they now each contain a modules directory. It is here that you will find your SE Linux configurations.

Multi-layer Security (MLS)

Multi-layer security (MLS) adds a more refined security access method. MLS adds a security level value to resources. Only users with access to certain levels can access the corresponding files and applications. Within each level access can be further controlled with the use of categories. Categories work much like groups, allowing access only to users cleared for that category. Access becomes more refined, instead of an all or nothing situation.

Multi-category Security (MCS)

Multi-category security (MCS) extends SE Linux not only to use by administrators, but also by users. Users can set categories that restrict and control access to their files and applications. Though based on MLS, MCS uses only categories, not security levels. Users can select a category for a file, only the administrator can create a category and determine what users can access it. Though similar in concept to a ACL (access control list), it differs in that it makes use of the SE Linux security structure, providing user-level control enforced by SE Linux.

Policy Methods

Operating system services and components are categorized in SELinux by their type and their role. Rules controlling these objects can be type based or role based. Policies are implemented using two different kinds of rules, type enforcement (TE) and role-based access control (RBAC). Multi-layer security (MLS) is an additional method further restricting access by security level. Security context now feature both the role of an object such as a user, and that object's security level.

Type Enforcement

With a type structure, the operating system resources are partitioned off into types, with each object assigned a type. Processes are assigned to domains. Users are restricted to certain domains, allowed to use only objects accessible in those domains.

Role-Based Access Control

A role-based approach focuses on controlling users. Users are assigned roles, which define what resources they can use. In a standard system, file permissions, such as those for groups, can control user access to files and directories. With roles, permissions become more flexible and refined. Certain users can have more access to services than others.

SELinux Users

Users will retain the permissions available on a standard system. In addition, SELinux can set up its own controls for a given user, defining a role for that user. General security identities created by SELinux include:

system_u The user for system processes

user_u To allow normal users to use a service

root For the root user

Policy Files

Policies are implemented in policy files. These are binary files compiled from source files. For a pre-configured targeted policy file, the policy binary files are in policy subdirectories in the **/etc/selinux** configuration directory, **/etc/selinux/targeted**. For example, the policy file for the targeted policy is

```
/etc/selinux/targeted/policy/policy.20
```

The targeted development files that hold the interface files are installed at **/usr/share/selinux**.

```
/usr/share/selinux/targeted
```

You can use the development files to create your own policy modules that you can then load.

SELinux Configuration

Configuration for general SELinux server settings is carried out in the **/etc/selinux/config** directory. Currently there are only two settings available: the state and the policy. You set the SELINUX variable to the state, such as enforcing or permissive, and the SELINUXTYPE variable to the kind of policy you want. These correspond to the firewall-config SELinux settings for disabled and enforcing, as well as the policy to use, such as targeted. A sample config file is shown here:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - SELinux is fully disabled.
SELINUX=permissive
# SELINUXTYPE= type of policy in use. Possible values are:
#     targeted - Only targeted network daemons are protected.
#     strict - Full SELinux protection.
SELINUXTYPE=targeted
```

SELinux Policy Rules

Policy rules can be made up of either type (Type Enforcement) or rbac (Role Based Access Control) statements, along with security levels (Multi-level Security). A type statement can be a type or attribute declaration, or a transition, change, or assertion rule. The rbac statements can be role declarations or dominance or allow roles. A security level specifies a number corresponding to the level of access permitted. Policy configuration can be difficult, using extensive and complicated rules. For this reason, many rules are implemented using M4 macros in **fi** files that will in turn generate the appropriate rules (Sendmail uses M4 macros in a similar way). You will find these rules in files in the SE Linux reference policy source code package which you need to download and install. The reference policy package is **selinux-policy**, located in the **RPMS** source code repository of Fedora. See the following section on SELinux configuration files.

Type and Role Declarations

A type declaration starts with the keyword **type**, followed by the type name (identifier) and any optional attributes or aliases. The type name will have a _t suffix. Standard type definitions are included for objects such as files. The following is a default type for any file, with attributes file_type and sysadmfile:

```
type file_t, file_type, sysadmfile;
```

The root will have its own type declaration.

```
type root_t, file_type, sysadmfile;
```

Specialized directories such as the boot directory will also have their own type.

```
type boot_t, file_type, sysadmfile;
```

More specialized rules are set up for specific targets like the Amanda server. The following example is the general type definition for amanda_t objects, those objects used by the Amanda backup server, as listed in the targeted policy's **src/program/amanda.te** file.

```
type amanda_t, domain, privlog, auth, nscd_client_domain ;
```

A role declaration determines the roles that can access objects of a certain type. These rules begin with the keyword **role** followed by the role and the objects associated with that role. In this example, the amanda objects (amanda_t) can be accessed by a user or process with the system role (system_r).

```
role system_r types amanda_t;
```

A more specific type declaration is provided for executables, such as the following for the Amanda server (amanda_exec_t). This defines the Amanda executable as a system administration–controlled executable file.

```
type amanda_exec_t, file_type, sysadmfile, exec_type;
```

Associated configuration files often have their own rules.

```
type amanda_config_t, file_type, sysadmfile;
```

In the targeted policy an general unconfined type is created that user and system roles can access, giving complete unrestricted access to the system. More specific rules will restrict access to certain targets like the Web server.

```
type unconfined_t, domain, privuser, privhome, privrole, privowner, admin,
auth_write, fs_domain, privmem;
role system_r types unconfined_t;
role user_r types unconfined_t;
role sysadm_r types unconfined_t;
```

Types are also set up for the files created in the user home directory.

```
type user_home_t, file_type, sysadmfile, home_type;
type user_home_dir_t, file_type, sysadmfile, home_dir_type;
```

File Contexts

File contexts associate specific files with security contexts. The file or files are listed first, with multiple files represented with regular expressions. Then the role, type, and security level are specified. The following creates a security context for all files in the **/etc** directory (configuration files). These are accessible from the system user (system_u) and are objects of the etc_t type with a security level of 0, s0.

```
/etc(/.*)?              system_u:object_r:etc_t:s0
```

Certain files can belong to other types, for instance, the **resolve.conf** configuration file belongs to the net_conf type.

```
/etc/resolv/.conf.*     --     system_u:object_r:net_conf_t:s0
```

Certain services will have their own security contexts for their configuration files.

```
/etc/amanda(/.*)?            system_u:object_r:amanda_config_t:s0
```

File contexts are located in the **file_contexts** file in the policy's contexts directory, such as **/etc/selinux/targeted/contexts/files/file_contexts**. The version used to create or modify the policy is located in the policy modules active directory, as in **targeted/modules/active/file_contexts**.

User Roles

User roles define what roles a user can take on. Such as role begins with the keyword **`user`** followed by the user name, then the keyword **`roles`**, and finally the roles it can use. You will find these rules in the selinux reference policy source code files. The following example is a definition of the system_u user:

```
user system_u roles system_r;
```

If a user can have several roles, then they are listed in brackets. The following is the definition of the standard user role in the targeted policy, which allows users to take on system administrative roles.

```
user user_u roles { user_r sysadm_r system_r };
```

The strict policy lists only the user_r role.

```
user user_u roles { user_r };
```

Access Vector Rules: allow

Access vector rules are used to define permissions for objects and processes. The **allow** keyword is followed by the object or process type, and then the types it can access or be accessed by and the permissions used. The following allows processes in the amanda_t domain to search the Amanda configuration directories (any directories of type amanda_config_t).

```
allow amanda_t amanda_config_t:dir search;
```

The following example allows Amanda to read the files in a user home directory:

```
allow amanda_t user_home_type:file { getattr read };
```

The next example allows Amanda to read, search, and write files in the Amanda data directories:

```
allow amanda_t amanda_data_t:dir { read search write };
```

Role Allow Rules

Roles can also have allow rules. Though they can be used for domains and objects, they are usually used to control role transitions, specifying whether a role can transition to another role. These rules are listed in the rbac configuration file. The following entry allows the user to transition to a system administrator role:

```
allow user_r sysadm_r;
```

Transition and Vector Rule Macros

The type transition rules set the type used for rules to create objects. Transition rules also require corresponding access vector rules to enable permissions for the objects or processes. Instead of creating separate rules, macros are used that will generate the needed rules. The following example sets the transition and access rules for user files in the home directory using the file_type_auto_trans macro:

```
file_type_auto_trans(privhome, user_home_dir_t, user_home_t)
```

The next example sets the Amanda process transition and access rules for creating processes:

```
domain_auto_trans(inetd_t, amanda_inetd_exec_t, amanda_t)
```

Constraint Rules

Restrictions can be further placed on processes such as transitions to ensure greater security. These are implemented with constraint definitions in the constraints file. Constraint rules are often applied to transition operations, such as requiring that, in a process transition, user identities remain the same, or that process 1 be in a domain that has the privuser attribute and process 2 be in a domain with the userdomain attribute. The characters **u**, **t**, and **r** refer to user, type, and role.

```
constrain process transition
    ( u1 == u2 or ( t1 == privuser and t2 == userdomain )
```

SE Linux Policy Configuration Files

Configuration files are normally changed using **.te** an **.fc** files. These are missing from the module headers in **/usr/share/selinux**. If you are adding a module you will need to create the .te and .fc files for it. Then you can create a module and add it as described in the next section. If you want to create or modify your own policy, you will need to download and install the source code files for the SE Linux reference policy as described the section after that. The reference policy code holds the complete set of **.te** and **.fc** configuration files.

Compiling SE Linux Modules

Instead of compiling the entire source each time you want to make a change, you can just compile a module for the area you changed. The modules directory holds the different modules. Each module is built from a corresponding .te file. The **checkmodule** command is used to create a **.mod** module file from the **.te** file, and then the **semanage_module** command is use to create the loadable **.pp** module file as well as a **.fc** file context file. As noted in the SE linux documentation, if you need to just change the configuration for **rsyslogd**, you would first use the following to create a **rsyslogd.mod** file using **rsyslogd.te**. The **-M** option specifies support for MLS security levels.

```
checkmodule -M -m rsyslogd.te  -o rsyslogd.mod
```

Then use the **semanage_module** command to create a **rsyslogd.pp** file from the rsyslogd.mod file. The **-f** opton specifies the file context file.

```
semanage_module -m rsyslogd.mod  -o rsyslogd.pp -f rsyslogd.fc
```

To add the module you use semodule and the -i option. You can check if a module is loaded with the **-l** option.

```
semodule -i rsyslogd.pp
```

Changes to the base policy are made to the **policy.conf** file which is compiled into the **base.pp** module.

Using SE Linux Source Configuration

To perform you own configuration, you will have to download and install the source code file **selinux-policy** from the **SRPMS** directory (**source**). The **.te** files used for configuring SE Linux are not part of the SE Linux binary packages.

Once the SRPMS file is downloaded, you would then have to use an rpm command with the **-i** option in a terminal window to install the source code.

```
rpm -i selinux-policy-2.5.13-18.fc10.srpms
```

The compressed archive of the source, a tgz file, along with various policy configuration files, will be installed to **/usr/src/redhat/SOURCES**. (be sure you have already installed **rpm-build**, it is not installed by default). Documentation for using the source files are in another source SRPMS file called **selinux-doc**.

You use an **rpmbuild** operation to extract the file to the **serefpolicy** directory in **/usr/src/redhat/BUILD**. Within this directory you will find a **policy/modules** subdirectory. There, organized into several directories like admin and apps you will find the **.tc**, **.fc**, and **.if** configuration files.

```
rpmbuild -bp  security-policy.spec
```

Then you change to the **/usr/src/redhat/BUILD/seref-policy** directory and run the following command to install selinux source to **/etc/selinux/serefpolicy/src**.

```
make install-src
```

The rules are held in configuration files located in various subdirectories in a policy's **src** directory. Within this directory you will find a **policy/modules** subdirectory. There, organized into several directories like admin and apps you will find the **.tc**, **.fc**, and .if configuration files.

You will have configuration files for both type enforcement and security contexts. Type enforcement files have the extension **.te**, whereas security contexts have an **.sc** extension.

Reflecting the fine-grained control that SELinux provides, you have numerous module configuration files for the many kinds of objects and processes on your system. The primary configuration files and directories are listed in Table 12-3, but several expand to detailed listing of subdirectories and files.

InterfaceFiles

File *interface* files allow management tools to generate policy modules. They define interface macros for your current policy. The refpolicy SE Linux source file will hold .if files for each module, along with .te and .fc files. Also, .if files in the **/usr/share/selinux/devel** directory which can be used to generate modules.

Types Files

In the targeted policy, the modules directory that defines types holds a range of files, including **nfs.te** and **network.te** configuration files. Here you will find type declarations for the different kinds of objects on your system. The .te files are no longer included with your standard SE Linux installation. Instead you have to download and install the serefpolicy source package. This is the original source and allows you to completely reconfigure your SE Linux policy, instead of managing modules with management tools like **semanage**. The modules directory will hold .te files for each module listing their TE rules.

Module Files

Module files are located among several directories in the policy/modules directory. Here you will find three corresponding files for each application or service. There will be a **.te** file that contains the actual type enforcement rules, an **.fi**, for interface, file that allows other applications to interact with the module, and the **.fc** files that define the file contexts.

Security Context Files

Security contexts for different files are detailed in security context files. The **file_contexts** file holds security context configurations for different groups, directories, and files. Each configuration file has an **.fc** extension. The **types.fc** file holds security contexts for various system files and directories, particularly access to configuration files in the **/etc** directory. In the SE Linux source, each module will have its own .fc file, along with corresponding .te and .if files. The **distros.fc** file defines distribution-dependent configurations. The **homedir_template** file defines

security contexts for dot files that may be set up in a user's home directory, such as **.mozilla**, **.gconf**, and **.java**.

A modules directory has file context files for particular applications and services. For example, **apache.fc** has the security contexts for all the files and directories used by the Apache Web server, such as **/var/www** and **/etc/httpd**.

User Configuration: Roles

Global user configuration is defined in the policy directory's **users** file. Here you find the user definitions and the roles they have for standard users (user_u) and administrators (admin_u). To add your own users, you use the **local.users** file. Here you will find examples for entering your own SELinux users. Both the strict and targeted policies use the general user_u SELinux identity for users. To set up a separate SELinux identity for a user, you would define that user in the **local.users** file.

The **rbac** file defines the allowed roles one role can transition to. For example, can the user role transition to an system administration role? The targeted policy has several entries allowing a user to freely transition to an administrator, and vice versa. The strict policy has no such definitions.

Role transitions are further restricted by rules in the **constraints** file. Here the change to other users is controlled, and changing object security contexts (labeling) is restricted.

Policy module tools

To create a policy module and load it, you use several policy module tools. First the **checkmodule** command is used to create **.mod** file from a **.te** file. Then the **semodule_package** tool takes the **.mod** file and any supporting **.fc** file and generates a module policy package file, **.pp**. Finally the **semodule** tool can take the policy package file and install it as part of your SE Linux policy.

Application Configuration: appconfig

Certain services and applications are security aware and will request default security contexts and types from SELinux (see also the upcoming section "Runtime Security Contexts"). The configuration is kept files located in the **policy/config/appconfig-*** directory. The **default_types** file holds type defaults; **default_contexts** holds default security contexts. The **initrc_context** file has the default security context for running **/etc/rc.d** scripts. A special **root_default_contexts** file details how the root user can be accessed. The **removable_context** file holds the security context for removable devices, and **media** lists media devices, such as cdrom for CD-ROMs. Runtime values can also be entered in corresponding files in the policy contexts directory, such as **/etc/selinux/targeted/contexts**.

Directories and Files	Description
assert.te	Access vector assertions
config/appconfig-*	Application runtime configuration files
policy/booleans.conf	Tunable features
file_contexts	Security contexts for files and directories
policy/flask	Flask configuration
policy/mcs	Multi-category security (MCS) configuration
doc	Policy documentation support
policy/modules	Security policy modules
policy/modules.conf	Module list and use
policy/modules/admin	Administration modules
policy/modules/apps	Application modules
policy/modules/kernel	Kernel modules
policy/modules/services	Services and server modules
policy/modules/system	System modules
policy/rolemap	User domain types and roles
policy/users	General users definition
config/local.users	Define your own SELinux users
policy/constraints	Additional constraints for role transition and object access
policygentool	Script to generate policies
policy/global_tunables	Defines policy tunables for customization
policy/mls	Multilevel security (MLS) configuration

Table 12-3: SELinux Policy Configuration Files

Creating an SELinux Policy: make and checkpolicy

If you want to create an entirely new policy, you use the SE Linux reference policy source, **/etc/selinux/serefpolicy**. Once you have configured your policy you can create it with the **`make policy`** and **`checkpolicy`** commands. The **`make policy`** command generates a policy.conf file for your configuration files, which **`checkpolicy`** can then use to generate a policy binary file. A policy binary file will be created in the **policy** subdirectory with a numeric extension for the policy version, such as **policy.20**.

You will have to generate a new policy **policy.conf** file. To do this you enter the following command in the policy src directory, which will be **/etc/selinux/serefpolicy/src/policy**.

```
make policy
```

Then you can use **checkpolicy** to create the new policy.

Instead of compiling the entire source each time you want to make a change, you can just compile a module for the area you changed. In the previous SE Linux version you always had to recompile the entire policy every time you made a change. The modules directory holds the different modules. Each module is built from a corresponding **.te** file. The **checkmodule** command is used to create a .mod module file from the **.te** file, and then the **semanage_module** command is use to create the loadable policy package **.pp** module file. As noted in the SE Linux documentation, if you need to just change the configuration for **rsyslogd**, you would first use the following to create an **rsyslogd.mod** file using **rsyslogd.te**. The **-M** option specifies support for MLS security levels.

```
checkmodule -M  -m rsyslogd.te  -o rsyslogd.mod
```

Then use the **semanage_module** command to create an **rsyslogd.pp** file from the rsyslogd.mod file. The **-f** option specifies the file context file.

```
semanage_module -m rsyslogd.mod  -o rsyslogd.pp -f rsyslogd.fc
```

To add the module you use semodule and the **-i** option. You can check if a module is loaded with the **-l** option.

```
semodule -i rsyslogd.pp
```

Changes to the base policy are made to the **policy.conf** file which is compiled into the **base.pp** module.

To perform you own configuration, you will now have to download the source code files. The **.te** files used for configuring SE Linux are no longer part of the SE Linux binary packages. Once installed the source will be in the **sefepolicy** directory in **/etc/selinux**.

For **fixfiles**, you use:

```
fixfiles relabel
```

SELinux: Administrative Operations

There are several tasks you can perform on your SELinux system without having to recompile your entire configuration. Security contexts for certain files and directories can be changed as needed. For example, when you add a new file system, you will need to label it with the appropriate security contexts. Also, when you add users, you may need to have a user given special attention by the system. For detecting and fixing problems you can use **setroubleshoot**.

Using Security Contexts: fixfiles, setfiles, restorecon, and chcon

Several tools are available for changing your objects' security contexts. The **fixfiles** command can set the security context for file systems. You use the **relabel** option to set security contexts, and the **check** option to see what should be changed. The **fixfiles** tool is a script that uses **setfiles** and **restorecon** to make actual changes.

The **restorecon** command will let you restore the security context for files and directories, but **setfiles** is the basic tool for setting security contexts. It can be applied to individual files or directories. It is used to label the file when a policy is first installed.

With **chcon**, you can change the permissions of individual files and directories, much as **chmod** does for general permissions.

Adding New Users

If a new user needs no special access, you can generally just use the generic SELinux user_u identity. If, however, you need to allow the user to take on roles that would otherwise be restricted, such as a system administrator role in the strict policy, you need to configure the user accordingly. To do this, you add the user to the **local.users** file in the policy users directory, as in **/etc/selinux/targeted/policy/users/local.users**. Note that this is different from the **local.users** file in the src directory which compiled directly into the policy. The user rules have the syntax

```
user username roles { rolelist };
```

The following example adds the sysadm role to the **george** user:

```
user george roles { user_r sysadm_r };
```

Once the role is added, you have to reload the policy.

```
make reload
```

Runtime Security Contexts and Types: contexts

Several applications and services are security aware and will need default security configuration information such as security contexts. Runtime configurations for default security contexts and types are kept in files located in the policy context directory, such as **/etc/selinux/targeted/contexts**. Types files will have the suffix **_types**, and security context files will use **_context**. For example, the default security context for removable files is located in the **removable_context** file. The contents of that file are shown here.

```
system_u:object_r:removable_t
```

The **default_context** file is used to assign a default security context for applications. In the strict policy it is used to control system admin access, providing it where needed, for instance, during the login process.

The following example sets the default roles for users in the login process:

```
system_r:local_login_t user_r:user_t
```

This would allow users to log in either as administrators or as regular users.

```
system_r:local_login_t sysadm_r:sysadm_t user_r:user_t
```

This next example is for remote user logins, where system administration is not included:

```
system_r:remote_login_t user_r:user_t staff_r:staff_t
```

The **default_types** file defines default types for roles. This file has role/type entries, and when a transition takes place to a new role, the default type specified here is used. For example, the default type for the sysadm_r role is sysadm_t.

```
sysadm_r:sysadm_t
user_r:user_t
```

Of particular interest is the **initrc_context** file, which sets the context for running the system scripts in the **/etc/rc.d** directory. In the targeted policy these are open to all users.

```
user_u:system_r:unconfined_t
```

In the strict policy these are limited to the system user.

```
system_u:system_r:initrc_t
```

users

Default security contexts may also need to be set up for particular users such as the root user. In the **sesuers** file you will find a root file that lists roles, types, and security levels the root user can take on, such as the following example for the su operation:

```
sysadm_r:sysadm_su_t  sysadm_r:sysadm_t  staff_r:staff_t  user_r:user_t
```

context/files

Default security contexts for your files and directories are located in the **contexts/files** directory. The **file_contexts** directory lists the default security contexts for all your files and directories as set up by your policy. The **file_context.homedirs** directory sets the file contexts for user home directory files as well as the root directory, including dot configuration files like **.mozilla** and **.gconf**. The media file sets the default context for media devices such as CD-ROMs and disks.

```
cdrom system_u:object_r:removable_device_t
floppy system_u:object_r:removable_device_t
disk system_u:object_r:fixed_disk_device_t
```

fedora

13. SSH, Kerberos, and IPsec

To protect remote connections from hosts outside your network, transmissions can be encrypted (see Table 13-1). For Linux systems, you can use the Secure Shell (SSH) suite of programs to encrypt and authenticate transmissions, preventing them from being read or modified by anyone else, as well confirming the identity of the sender. The SSH programs are meant to replace the remote tools such as **rsh** and **rcp**, which perform no encryption and include security risks such as transmitting passwords in clear text. SSH is available on the Fedora repository. It is considered an integral part of the Fedora distribution.

User authentication can be controlled for certain services by Kerberos servers. Kerberos authentication provides another level of security whereby individual services can be protected, allowing use of a service only to users who are cleared for access. Kerberos is provided as part of the universe repository.

Website	Description
www.openssh.org	OpenSSH open source version of SSH
www.ssh.com	SSH Communications Security, commercial SSH version
http://web.mit.edu/kerberos	Kerberos authentication

Table 13-1: SSH and Kerberos Resources

The Secure Shell: OpenSSH

Although a firewall can protect a network from attempts to break into it from the outside, the problem of securing legitimate communications to the network from outside sources still exists. A particular problem is one of users who want to connect to your network remotely. Such connections could be monitored, and information such as passwords and user IDs used when the user logs in to your network could be copied and used later to break in. One solution is to use SSH for remote logins and other kinds of remote connections such as FTP transfers. SSH encrypts any communications between the remote user and a system on your network.

Two different implementations of SSH currently use what are, in effect, two different and incompatible protocols. The first version of SSH, known as SSH1, uses the original SSH protocol. Version 2.0, known as SSH2, uses a completely rewritten version of the SSH protocol. Encryption is performed in different ways, encrypting different parts of a packet. SSH1 uses server and host keys to authenticate systems, whereas SSH2 uses only host keys. Furthermore, certain functions, such as sftp, are supported only by SSH2.

Note: A commercial version of SSH is available from SSH Communications Security, whose website is www.ssh.com. SSH Communications Security provides an entirely commercial version called SSH Tectia, designed for enterprise and government use. The older noncommercial SSH package is still freely available, which you can download and use.

The SSH protocol has become an official Internet Engineering Task Force (IETF) standard. A free and open source version is developed and maintained by the OpenSSH project, currently supported by the OpenBSD project. OpenSSH is the version supplied with most Linux distributions, including Fedora. You can find out more about OpenSSH at www.openssh.org, where you can download the most recent version, though your distribution will provide current RPM versions.

SSH Encryption and Authentication

SSH secures connections by both authenticating users and encrypting their transmissions. The authentication process is handled with public key encryption. Once authenticated, transmissions are encrypted by a cipher agreed upon by the SSH server and client for use in a particular session. SSH supports multiple ciphers. Authentication is applied to both hosts and users. SSH first authenticates a particular host, verifying that it is a valid SSH host that can be securely communicated with. Then the user is authenticated, verifying that the user is who they say they are.

SSH uses strong encryption methods, and their export from the United States may be restricted. Currently, SSH can deal with the following kinds of attacks:

- IP spoofing, where a remote host sends out packets that pretend to come from another, trusted host
- IP source routing, where a host can pretend an IP packet comes from another, trusted host
- DNS spoofing, where an attacker forges name server records
- Interception of clear-text passwords and other data by intermediate hosts
- Manipulation of data by people in control of intermediate hosts
- Attacks based on listening to X authentication data and spoofed connections to the X11 server

Encryption

The public key encryption used in SSH authentication makes use of two keys: a public key and a private key. The *public key* is used to encrypt data, while the *private key* decrypts it. Each host or user has its own public and private keys. The public key is distributed to other hosts, who can then use it to encrypt authentication data that only the host's private key can decrypt. For example, when a host sends data to a user on another system, the host encrypts the authentication data with a public key, which it previously received from that user. The data can be decrypted only by the user's corresponding private key. The public key can safely be sent in the open from one host to another, allowing it to be installed safely on different hosts. You can think of the process as taking place between a client and a server. When the client sends data to the server, it first encrypts the data using the server's public key. The server can then decrypt the data using its own private key.

It is recommended that SSH transmissions be authenticated with public-private keys controlled by passphrases. Unlike PGP, SSH uses public-key encryption for the authentication process only. Once authenticated, participants agree on a common cipher to use to encrypt transmission. Authentication will verify the identity of the participants. Each user who intends to use SSH to access a remote account first needs to create the public and private keys along with a passphrase to use for the authentication process. A user then sends their public key to the remote account they want to access and installs the public key on that account. When the user attempts to access the remote account, that account can then use the user's public key to authenticate that the user is who they claim to be. The process assumes that the remote account has set up its own SSH private and public key. For the user to access the remote account, they will have to know the remote account's SSH passphrase. SSH is often used in situations where a user has two or more accounts located on different systems and wants to be able to securely access them from each other.

In that case, the user already has access to each account and can install SSH on each, giving each its own private and public keys along with their passphrases.

Authentication

The mechanics of authentication in SSH version 1 and version 2 differ slightly. However, the procedure on the part of users is the same. Essentially, a user creates both public and private keys. For this you use the **`ssh-keygen`** command. The user's public key then has to be distributed to those users that the original user wants access to. Often this is an account a user has on another host. A passphrase further protects access. The original user will need to know the other user's passphrase to access it.

SSH version 1 uses RSA authentication. When a remote user tries to log in to an account, that account is checked to see if it has the remote user's public key. That public key is then used to encrypt a challenge (usually a random number) that can be decrypted only by the remote user's private key. When the remote user receives the encrypted challenge, that user decrypts the challenge with its private key. SSH version 2 can use either RSA or DSA authentication. The remote user will first encrypt a session identifier using its private key, signing it. The encrypted session identifier is then decrypted by the account using the remote user's public key. The session identifier has been previously set up by SSH for that session.

SSH authentication is first carried out with the host, and then with users. Each host has its own host keys, public and private keys used for authentication. Once the host is authenticated, the user is queried. Each user has their own public and private keys. Users on an SSH server who want to receive connections from remote users will have to keep a list of those remote user's public keys. Similarly, an SSH host will maintain a list of public keys for other SSH hosts.

SSH Packages, Tools, and Server

SSH is implemented on Linux systems with OpenSSH. The full set of OpenSSH packages includes the OpenSSH meta-package (ssh), the OpenSSH server (openssh-server), and the OpenSSH client (openssh-clients). These packages also require OpenSSL (openssl), which installs the cryptographic libraries that SSH uses.

The SSH tools are listed in Table 13-2. They include several client programs such as **scp**, **ssh**, as well as the **ssh** server. The **ssh** server (**sshd**) provides secure connections to anyone from the outside using the **ssh** client to connect. Several configuration utilities are also included, such as ssh-add, which adds valid hosts to the authentication agent, and ssh-keygen, which generates the keys used for encryption.

For version 2, names of the actual tools have a *2* suffix. Version 1 tools have a *1* as their suffix. During installation, however, links are set for each tool to use only the name with the suffix. For example, if you have installed version 2, there is a link called **scp** to the **scp2** application. You can then use the link to invoke the tool. Using **scp** starts **scp2**. Table 13-2 specifies only the link names, as these are the same for each version. Remember, though, some applications, such as sftp, are available only with version 2.

Application	Description
ssh	SSH client
sshd	SSH server (daemon)
sftp	SSH FTP client, Secure File Transfer Program. Version 2 only. Use **?** to list sftp commands(SFTP protocol)
sftp-server	SSH FTP server. Version 2 only (SFTP protocol)
scp	SSH copy command client
ssh-keygen	Utility for generating keys. **-h** for help
ssh-keyscan	Tool to automatically gather public host keys to generate ssh_known_hosts files
ssh-add	Adds RSD and DSA identities to the authentication agent
ssh-agent	SSH authentication agent that holds private keys for public key authentication (RSA, DSA)
ssh-askpass	X Window System utility for querying passwords, invoked by **ssh-add** (openssh-askpass)
ssh-askpass-gnome	GNOME utility for querying passwords, invoked by **ssh-add**
ssh-signer	Signs host-based authentication packets. Version 2 only. Must be suid root (performed by installation)
slogin	Remote login (version 1)

Table 13-2: SSH Tools

You can configure the Openssh server (sshd) to start up automatically using **system-config-services** (System | Administration | Services) and selecting **sshd**. You can start, stop, and restart the server manually with the **/etc/init.d/sshd** script.

```
sudo /etc/init.d/sshd restart
```

You have to configure your firewall to allow access to the ssh service. Open **system-config-firewall** (System | Administration | Firewall) and select the Trusted Services entry to display the list of trusted services. Scrolling down, you will see an entry for SSH. Click on its checkbox, if not already checked. The services is set ups to operate using the TCP protocol on port 22, tcp/22 You can configure a different port to use in the **/etc/ssh/shhd_config** file if you wish and then open that port on the firewall.

If you are managing your IPTables firewall directly, you could manage access directly by adding the following IPtables rule. This accepts input on port 22 for TCP/IP protocol packages.

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

SSH Setup

Using SSH involves creating your own public and private keys and then distributing your public key to other users you want to access. These can be different users or simply user accounts of your own that you have on remote systems. Often people remotely log in from a local client to an account on a remote server, perhaps from a home computer to a company computer. Your home

computer would be your client account, and the account on your company computer would be your server account. On your client account, you need to generate your public and private keys and then place a copy of your public key in the server account. You can do this by simply e-mailing the key file or copying the file from a floppy disk. Once the account on your server has a copy of your client user's public key, you can access the server account from your client account. You will be also prompted for the server account's passphrase. You will have to know this to access that account. Figure 13-1 illustrates the SSH setup that allows a user **george** to access the account **cecelia**.

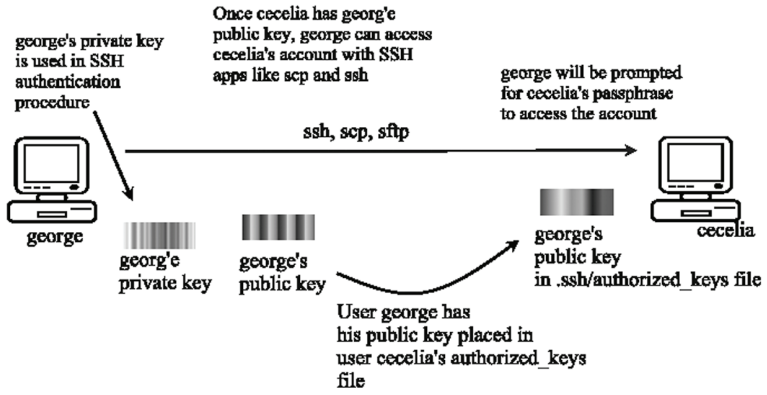


Figure 13-1: SSH setup and access

To allow you to use SSH to access other accounts:

- You must create public and private keys on your account along with a passphrase. You will need to use this passphrase to access your account from another account.
- You must distribute your public key to other accounts you want to access, placing them in the **.ssh/authorized_keys** file.
- Other accounts also have to set up public and private keys along with a passphrase.
- You must know the other account's passphrase to access it.

Creating SSH Keys with ssh-keygen

You create your public and private keys using the **`ssh-keygen`** command. You need to specify the kind of encryption you want to use. You can use either DSA or RSA encryption. Specify the type using the **`-t`** option and the encryption name in lowercase (**`dsa`** or **`rsa`**). In the following example, the user creates a key with the RSA encryption:

```
ssh-keygen -t rsa
```

The **`ssh-keygen`** command prompts you for a passphrase, which it will use as a kind of password to protect your private key. The passphrase should be several words long. You are also prompted to enter a filename for the keys. If you do not enter one, SSH will use its defaults. The public key will be given the extension **.pub**. The **`ssh-keygen`** command generates the public key

and places it in your **.ssh/id_dsa.pub** or **.ssh/id_dsa.pub** file, depending on the type of key you specified; it places the private key in the corresponding **.ssh/id_dsa** or **.ssh/id_rsa.pub** file.

If you need to change your passphrase, you can do so with the **`ssh-keygen`** command and the **`-p`** option. Each user will have their own SSH configuration directory, called **.ssh**, located in their own home directory. The public and private keys, as well as SSH configuration files, are placed here. If you build from the source code, the **`make install`** operation will automatically run **`ssh-keygen`**. Table 13-3 lists the SSH configuration files.

File	Description
$HOME/.ssh/known_hosts	Records host keys for all hosts the user has logged in to (that are not in **/etc/ssh/ssh_known_hosts**).
$HOME/.ssh/random_seed	Seeds the random number generator.
$HOME/.ssh/id_rsa	Contains the RSA authentication identity of the user.
$HOME/.ssh/ id_dsa	Contains the DSA authentication identity of the user.
$HOME/.ssh/id_rsa.pub	Contains the RSA public key for authentication. The contents of this file should be added to **$HOME/.ssh/authorized_keys** on all machines where you want to log in using RSA authentication.
$HOME/.ssh/id_dsa.pub	Contains the DSA public key for authentication.
$HOME/.ssh/config	the per-user configuration file.
$HOME/.ssh/authorized_keys	Lists the RSA or DSA keys that can be used for logging in as this user.
/etc/ssh/ssh_known_hosts	Contains the system-wide list of known host keys.
/etc/ssh/ssh_config	Contains the system-wide configuration file. This file provides defaults for those values not specified in the user's configuration file.
/etc/ssh/sshd_config	Contains the SSH server configuration file.
/etc/ssh/sshrc	Contains the system default. Commands in this file are executed by ssh when the user logs in just before the user's shell (or command) is started.
$HOME/.ssh/rc	Contains commands executed by ssh when the user logs in just before the user's shell (or command) is started.

Table 13-3: SSH Configuration Files

Authorized Keys

A public key is used to authenticate a user and its host. You use the public key on a remote system to allow that user access. The public key is placed in the remote user account's **.ssh/authorized_keys** file. Recall that the public key is held in the **.ssh/id_dsa.pub** file. If a user wants to log in remotely from a local account to an account on a remote system, they would first place their public key in the **.ssh/authorized_keys** file in the account on the remote system they want to access. If the user **larisa** on **turtle.mytrek.com** wants to access the **aleina** account on

rabbit.mytrek.com, **larisa**'s public key from **/home/larisa/.ssh/id_dsa.pub** first must be placed in **aleina**'s **authorized_keys** file, **/home/aleina/.ssh/authorized_keys**. User **larisa** can send the key or have it copied over. A simple cat operation can append a key to the authorized key file. In the next example, the user adds the public key for **aleina** in the **larisa.pub** file to the authorized key file. The **larisa.pub** file is a copy of the **/home/larisa/.ssh/id_dsa.pub** file that the user received earlier.

```
cat larisa.pub >>  .ssh/authorized_keys
```

Note: You can also use seahorse to create and manage SSH keys.

Note: The **.ssh/identity** filename is used in SSH version 1; it may be installed by default on older distribution versions. SSH version 2 uses a different filename, **.ssh/id_dsa** or **.ssh/id_rsa**, depending on whether RSA or DSA authentication is used.

Loading Keys

If you regularly make connections to a variety of remote hosts, you can use the **ssh-agent** command to place private keys in memory where they can be accessed quickly to decrypt received transmissions. The **ssh-agent** command is intended for use at the beginning of a login session. For GNOME, you can use the openssh-askpass-gnome utility, invoked by **ssh-add**, which allows you to enter a password when you log in to GNOME. GNOME will automatically supply that password whenever you use an SSH client.

Although the **ssh-agent** command enables you to use private keys in memory, you also must specifically load your private keys into memory using the **ssh-add** command. **ssh-add** with no arguments loads your private key from your **.ssh/id_dsa** or **.ssh/id_rsa.pub** file. You are prompted for your passphrase for this private key. To remove the key from memory, use **ssh-add** with the **-d** option. If you have several private keys, you can load them all into memory. **ssh-add** with the **-l** option lists those currently loaded.

SSH Clients

SSH was originally designed to replace remote access operations, such as rlogin, rcp, and Telnet, which perform no encryption and introduce security risks such as transmitting passwords in clear text. You can also use SSH to encode X server sessions as well as FTP transmissions (sftp). The ssh-clients package contains corresponding SSH clients to replace these applications. With slogin or ssh, you can log in from a remote host to execute commands and run applications, much as you can with rlogin and rsh. With scp, you can copy files between the remote host and a network host, just as with rcp. With sftp, you can transfer FTP files secured by encryption.

ssh

With **ssh** you can remotely log in from a local client to a remote system on your network operating as the SSH server. The term *local client* here refers to one outside the network, such as your home computer, and the term *remote* refers to a host system on the network to which you are connecting. In effect, you connect from your local system to the remote network host. It is designed to replace rlogin, which performs remote logins, and rsh, which executes remote commands. With ssh, you can log in from a local site to a remote host on your network and then send commands to be executed on that host. The **ssh** command is also capable of supporting X Window System

connections. This feature is automatically enabled if you make an ssh connection from an X Window System environment, such as GNOME or KDE. A connection is set up for you between the local X server and the remote X server. The remote host sets up a dummy X server and sends any X Window System data through it to your local system to be processed by your own local X server.

The ssh login operation function is much like the **rlogin** command. You enter the **ssh** command with the address of the remote host, followed by a **-l** option and the login name (username) of the remote account you are logging in to. The following example logs in to the **aleina** user account on the **rabbit.mytrek.com** host:

```
ssh rabbit.mytrek.com -l aleina
```

You can also use the username in an address format with ssh, as in

```
ssh aleian@rabbit.mytrek.com
```

The following listing shows how the user **george** accesses the **cecelia** account on **turtle.mytrek.com**:

```
[george@turtle george]$ ssh turtle.mytrek.com -l cecelia
cecelia@turtle.mytrek.com's password:
[cecelia@turtle cecelia]$
```

A variety of options are available to enable you to configure your connection. Most have corresponding configuration options that can be set in the configuration file. For example, with the **-c** option, you can designate which encryption method you want to use, for instance, **idea**, **des**, **blowfish**, or **arcfour**. With the **-i** option, you can select a particular private key to use. The **-C** option enables you to have transmissions compressed at specified levels (see the **ssh** Man page for a complete list of options).

scp

You use **scp** to copy files from one host to another on a network. Designed to replace rcp, scp uses ssh to transfer data and employs the same authentication and encryption methods. If authentication requires it, scp requests a password or passphrase. The scp program operates much like rcp. Directories and files on remote hosts are specified using the username and the host address before the filename or directory. The username specifies the remote user account that scp is accessing, and the host is the remote system where that account is located. You separate the user from the host address with an @, and you separate the host address from the file or directory name with a colon. The following example copies the file **party** from a user's current directory to the user **aleina**'s **birthday** directory, located on the **rabbit.mytrek.com** host:

```
scp party aleina@rabbit.mytrek.com:/birthday/party
```

Of particular interest is the **-r** option (recursive) option, which enables you to copy whole directories. See the **scp** Man page for a complete list of options. In the next example, the user copies the entire **reports** directory to the user **justin**'s **projects** directory:

```
scp -r reports justin@rabbit.mytrek.com:/projects
```

In the next example, the user **george** copies the **mydoc1** file from the user **cecelia**'s home directory:

```
[george@turtle george]$ scp cecelia@turtle.mytrek.com:mydoc1  .
cecelia@turtle.mytrek.com's password:
mydoc1     0% |                            |   0 --:--
ETA
mydoc1   100% |****************************|  17 00:00
[george@turtle george]$
```

From a Windows system, you can also use **scp** clients such as **winscp**, which will interact with Linux scp-enabled systems.

sftp and sftp-server

With **sftp**, you can transfer FTP files secured by encryption. The **sftp** program uses the same commands as **ftp**. This client, which works only with ssh version 2, operates much like **ftp**, with many of the same commands. Use **sftp** instead of **ftp** to invoke the sftp client.

```
sftp download.fedora.redhat.com
```

To use the sftp client to connect to an FTP server, that server needs to be operating the sftp-server application. The ssh server invokes sftp-server to provide encrypted FTP transmissions to those using the sftp client. The sftp server and client use the SSH File Transfer Protocol (SFTP) to perform FTP operations securely.

Port Forwarding (Tunneling)

If, for some reason, you can connect to a secure host only by going through an insecure host, ssh provides a feature called port forwarding. With *port forwarding,* you can secure the insecure segment of your connection. This involves simply specifying the port at which the insecure host is to connect to the secure one. This sets up a direct connection between the local host and the remote host, through the intermediary insecure host. Encrypted data is passed through directly. This process is referred to as tunneling, creating a secure tunnel of encrypted data through connected servers.

You can set up port forwarding to a port on the remote system or to one on your local system. To forward a port on the remote system to a port on your local system, use **ssh** with the **-R** option, followed by an argument holding the local port, the remote host address, and the remote port to be forwarded, each separated from the next by a colon. This works by allocating a socket to listen to the port on the remote side. Whenever a connection is made to this port, the connection is forwarded over the secure channel, and a connection is made to a remote port from the local machine. In the following example, port 22 on the local system is connected to port 23 on the **rabbit.mytrek.com** remote system:

```
ssh -R 22:rabbit.mytrek.com:23
```

To forward a port on your local system to a port on a remote system, use the **ssh -L** option, followed by an argument holding the local port, the remote host address, and the remote port to be forwarded, each two arguments separated by a colon. A socket is allocated to listen to the port on the local side. Whenever a connection is made to this port, the connection is forwarded over the secure channel and a connection is made to the remote port on the remote machine. In the following example, port 22 on the local system is connected to port 23 on the **rabbit.mytrek.com** remote system:

```
ssh -L 22:rabbit.mytrek.com:23
```

You can use the LocalForward and RemoteForward options in your **.ssh/config** file to set up port forwarding for particular hosts or to specify a default for all hosts you connect to.

SSH Configuration

The SSH configuration file for each user is in their **.ssh/config** file. The **/etc/ssh/ssh_config** file is used to SSH client set site-wide defaults. In the configuration file, you can set various options, as listed in the **ssh_config** Man document. The configuration file is designed to specify options for different remote hosts to which you might connect. It is organized into segments, where each segment begins with the keyword **HOST**, followed by the IP address of the host. The following lines hold the options you have set for that host. A segment ends at the next **HOST** entry. Of particular interest are the **User** and **Cipher** options. Use the **User** option to specify the names of users on the remote system who are allowed access. With the **Cipher** option, you can select which encryption method to use for a particular host. Encryption methods include IDEA, DES (standard), triple-DES (3DES), Blowfish (128 bit), Arcfour (RSA's RC4), and Twofish. The following example allows access from **larisa** at **turtle.mytrek.com** and uses Blowfish encryption for transmissions:

```
Host turtle.mytrek.com
     User larisa
     Compression no
     Cipher blowfish
```

Most standard options, including ciphers are already listed as commented entries. Remove the # to activate.

```
#   Ciphers aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-
cbc,aes256-cbc
```

To specify global options that apply to any host you connect to, create a **HOST** entry with the asterisk as its host, **HOST ***. This entry must be placed at the end of the configuration file because an option is changed only the first time it is set. Any subsequent entries for an option are ignored. Because a host matches on both its own entry and the global one, its specific entry should come before the global entry. The asterisk (*****) and the question mark (**?**) are both wildcard matching operators that enable you to specify a group of hosts with the same suffix or prefix.

```
   Host *
     FallBackToRsh yes
     KeepAlive no
     Cipher 3des
```

The protocol option lets you specify what version of SSH to use, 1, 2, or both. By default, both protocols are acceptable, for backward compatibility. You can restrict use to just the more advanced and secure SSH2 protocol by changing the Protocol option to just 2.

```
Protocol 2
```

You use the **/etc/ssh/sshd**_config file to configure an SSH server. Here you will find server options like the port to use, password requirement, and PAM usage.

Kerberos

User authentication can further be controlled for certain services by Kerberos servers, discussed in this chapter. Kerberos authentication provides another level of security whereby individual services can be protected, allowing use of a service only to users who are cleared for access. Kerberos servers are all enabled and configured with **authconfig-gtk** (Authentication in the System | Administration menu).

Kerberos is a network authentication protocol that provides encrypted authentication to connections between a client and a server. As an authentication protocol, Kerberos requires a client to prove its identity using encryption methods before it can access a server. Once authenticated, the client and server can conduct all communications using encryption. Whereas firewalls protect only from outside attacks, Kerberos is designed to also protect from attacks from those inside the network. Users already within a network could try to break into local servers. To prevent this, Kerberos places protection around the servers themselves, rather than an entire network or computer. A free version is available from the Massachusetts Institute of Technology at **http://web.mit.edu/kerberos** under the MIT Public License, which is similar to the GNU Public License. The name *Kerberos* comes from Greek mythology and is the name of the three-headed watchdog for Hades. Be sure to check the **http://web.mit.edu/kerberos** site for recent upgrades and detailed documentation, including FAQs, manuals, and tutorials.

Fedora installs the Kerberos support libraries by default. You can install the Kerberos server and several Kerberos clients using the krb5 packages. The server is **krb5-server**, which will also install the kdc server and the krb5 libraries. The **krb5-workstation** will install needed support applications like **kadmin** for managing the Kerberos server. The **krb5-workstation-clients** package includes the Kerberos secured replacements for **rsh**, **rcp**, **telnet**, and the **ftp** client. Kerberos secured server plugins are also available for PAM, Python, LDAP, Perl, and Ruby. The LDAP plugin would be needed to operate a secure LDAP server.

Instead of running Kerberos server as a standalone server, you can run it on an on-demand basis using xinetd. This version of the Kerberos server is installed with the **kerberos-workstation-server** package.

Tip: The Kerberos V5 workstation clients package includes its own versions of network tools such as Telnet, RCP, FTP, and RSH. These provide secure authenticated access by remote users. The tools operate in the same way as their original counterparts. The package also contains a Kerberos version of the `su` administrative login command, `ksu`.

Kerberos Servers

The key to Kerberos is a Kerberos server through which all requests for any server services are channeled. The Kerberos server then authenticates a client, identifying the client and validating the client's right to use a particular server. The server maintains a database of authorized users. Kerberos then issues the client an encrypted ticket that the client can use to gain access to the server. For example, if a user needs to check their mail, a request for use of the mail server is sent to the Kerberos server, which then authenticates the user and issues a ticket that is then used to access the mail server. Without a Kerberos-issued ticket, no one can access any of the servers. Originally, this process required that users undergo a separate authentication procedure for each

server they wanted to access. However, users now only need to perform an initial authentication that is valid for all servers.

This process involves the use of two servers, an authentication server (AS) and a ticket-granting server (TGS). Together they make up what is known as the key distribution center (KDC). In effect, they distribute keys used to unlock access to services. The authentication server first validates a user's identity. The AS issues a ticket called the ticket-granting ticket (TGT) that allows the user to access the ticket-granting server. The TGS then issues the user another ticket to actually access a service. This way, the user never has any direct access of any kind to a server during the authentication process. The process is somewhat more complex than described. An authenticator using information such as the current time, a checksum, and an optional encryption key is sent along with the ticket and is decrypted with the session key. This authenticator is used by a service to verify your identity.

Note: You can view your list of current tickets with the `klist` command.

Authentication Process

The authentication server validates a user using information in its user database. Each user needs to be registered in the authentication server's database. The database will include a user password and other user information. To access the authentication server, the user provides the username and the password. The password is used to generate a user key with which communication between the AS and the user is encrypted. The user will have their own copy of the user key with which to decrypt communications. The authentication process is illustrated in Figure 13-2.

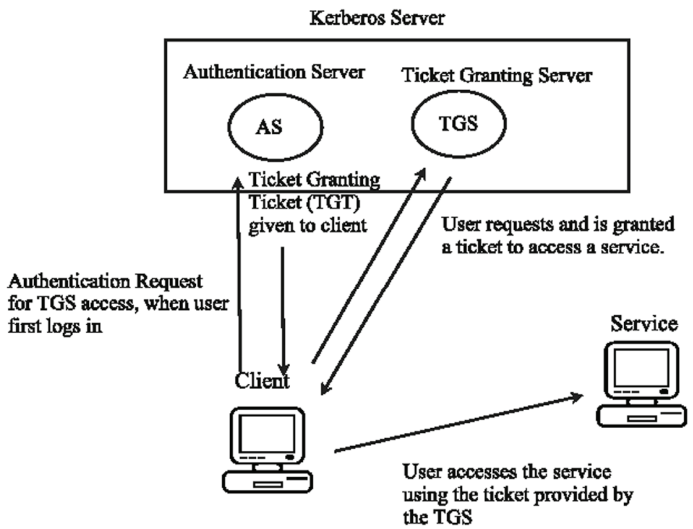


Figure 13-2: Kerberos authentication

Accessing a service with Kerberos involves the following steps:

1. The user has to be validated by the authentication server and granted access to the ticket-granting server with a ticket access key. You do this by issuing the **kinit** command, which will ask you enter your Kerberos username and then send it on to the authentication server (the Kerberos username is usually the same as your username).
2. The AS generates a ticket-granting ticket (TGT) with which to access the ticket-granting server (TGS). This ticket will include a session key that will be used to let you access the TGS. The TGT is sent back to you encrypted with your user key (password).
3. The **kinit** program then prompts you to enter your Kerberos password, which it then uses to decrypt the TGT. You can manage your Kerberos password with the **kpasswd** command.
4. Now you can use a client program such as a mail client program to access the mail server, for instance. When you do so, the TGT accesses the TGS, which then generates a ticket for accessing the mail server. The TGS generates a new session key for use with just the mail server. This is provided in the ticket sent to you for accessing the mail server. In effect, there is a TGT session key used for accessing the TGS, and a mail session key used for accessing the mail server. The ticket for the mail server is sent to you encrypted with the TGS session key.
5. The client then uses the mail ticket received from the TGS to access the mail server.
6. If you want to use another service such as FTP, when your FTP client sends a request to the TGS for a ticket, the TGS will automatically obtain authorization from the authentication server and issue an FTP ticket with an FTP session key. This kind of support remains in effect for a limited period of time, usually several hours, after which you again have to use **kinit** to undergo the authentication process and access the TGS. You can manually destroy any tickets you have with the **kdestroy** command.

Note: With Kerberos V5 (version 5), a Kerberos login utility is provided whereby users are automatically granted ticket-granting tickets when they log in normally. This avoids the need to use **kinit** to manually obtain a TGT.

Kerberized Services

Setting up a particular service to use Kerberos (known as Kerberizing) can be a complicated process. A Kerberized service needs to check the user's identity and credentials, check for a ticket for the service, and if one is not present, obtain one. Once they are set up, use of Kerberized services is nearly transparent to the user. Tickets are automatically issued and authentication carried out without any extra effort by the user. The **/etc/services** file should contain a listing of specific Kerberized services. These are services such as **kpasswd**, **kshell**, and **klogin** that provide Kerberos password, superuser access, and login services.

Kerberos also provides its own kerberized network tools for **ftp**, **rsh**, **rcp**, and **rlogin**. These are located at **/usr/kerberos/bin**, and most have the same name as the original network tools. In the Red Hat Fedora installation, the PATH variable, which contains the directories for all your commands, will list **/usr/kerberos/bin** before **/usr/bin**. This means that when you invoke ftp, you

are actually using the Kerberos version of ftp in **/usr/kerberos/bin**, instead of the original **ftp**, which is still located in **/usr/bin**. The same is true for **rlogin** and **rcp**. The original name for the kerberized network tools had a **k** prefix, and there are still links with some of these original names, such as **krlogin**, which links to the kerberos **rlogin**.

Kerberos Servers and Clients

Installing and configuring a Kerberos server is also a complex process. Carefully check the documentation for installing the current versions. Some of the key areas are listed here. In the Kerberos configuration file, **krb5.conf**, you can set such features as the encryption method used and the database name. When installing Kerberos, be sure to carefully follow the instructions for providing administrative access. To run Kerberos, you start the Kerberos server with **service** command and the **krb5kdc**, **kadmin**, and **krb524** scripts.

Tip: Check the Red Hat Linux Reference Manual for more detailed instructions on setting up Kerberos servers and clients on your system.

Consult the Red Hat Linux Reference Guide for detailed instructions on how to install and configure Kerberos on Red Hat. You will need to configure the server for your network, along with clients for each host (**krb5-server** package for servers and **krb5-workstation-clients** for clients).

To configure your server, you first specify your Kerberos realm and domain by manually replacing the lowercase **example.com** and the uppercase **EXAMPLE.COM** entries in the **/etc/krb5.conf** and **/var/kerberos/krb5kdc/kdc.conf** files with your own domain name. Maintain the same case for each entry. Realms are specified in uppercase, and simple host and domain names are in lowercase.

You will need to configure the server for your network, along with clients for each host (the **krb5-server** package for servers and **krb5-clients** for clients). To configure your server, you first specify your Kerberos realm and domain. You then create a database with the **kdb5_util** command and the **create** option. You will be prompted to enter a master key.

```
kdb5_util create -s
```

You then need to add a local principal, a local user with full administrative access from the host the server runs on. Start the **kadmin.local** tool and use the **addprincipal** command to add the local principal. You can then start your **krb5kdc**, **kadmin**, and **krb524** servers.

On each client host, use the **kadmin** tool with the **addprincipal** command to add a principal for the host. Also add a host principal for each host on your network with **host/** qualifier, as in **host/rabbit.mytrek.com**. You can use the **-randkey** option to specify a random key. Then save local copies of the host keys, using the **ktadd** command to save them in its **/etc/krb5.keytab** file. Each host needs to also have the same **/etc/krb5.conf** configuration file on its system, specifying the Kerberos server and the kdc host.

Note: When you configure Kerberos with the Authentication tool, you will be able to enter the realm, kdc server, and Kerberos server. Default entries will be displayed using the domain "example.com." Be sure to specify the realm in uppercase. A new entry for your realm will be made in the realms segment of the **/etc/krb5.conf**, listing the kdc and server entries you made.

Internet Protocol Security: IPsec

The Internet Security Protocol, IPsec, incorporates security for network transmission into the Internet Protocol (IP) directly. IPsec is integrated into the new IPv6 protocol (Internet Protocol version 6). It can also be used with the older IPv4 protocol. IPsec provides methods for both encrypting data and authenticating the host or network it is sent to. The process can be handled manually or automated using the IPsec **racoon** key exchange tool. With IPsec, the kernel can automatically detect and decrypt incoming transmissions, as well as encrypt outgoing ones. You can also use IPsec to implement virtual private networks, encrypting data sent over the Internet from one local network to another. Though IPsec is a relatively new security method, its integration into the Internet Protocol will eventually provide it wide acceptance. Check the IPsec Howto for a detailed explanation of IPsec implementation on Linux, **www.ipsec-howto.org**. The Red Hat Linux Enterprise Security Guide provides a helpful description on using IPsec on Red Hat system in its Virtual Private Network section. The Guide can be found at the Red Hat Linux Enterprise Documentation page on the Red Hat site, **www.redhat.com**.

Several projects currently provide development and implementation of IPsec tools (see Table 13-4). The original IPsec tools were provided by the KAME project, whose efforts have since been integrated into Linux and BSD Unix. The **ipsec-tools** package, which includes raccoon, is available on the Fedora repository. Other IPsec tool projects include the Open Secure/Wide Area Network project (Openswan) at **www.openswan.org**, which provides a Linux implementation of IPsec tools, and the VPN Consortium (VPNC) at **www.vpnc.org**, which supports Windows and Macintosh versions. Openswan is now included with Fedora. Documentation will be located at **/usr/doc/openswan-***version*. Detailed documentation is held in the **openswan-doc** package, which will be installed at **/usr/doc/openswan-doc-***version*.

Web Site	Project
www.openvpn.org	Open Secure/Wide Area Network project (Fedora)
www.openswan.org	Open Secure/Wide Area Network project (Fedora)
www.vpnc.org	VPN Consortium
www.ipsec-howto.org	IPsec Howto documentation

Table 13-4: VPN Resources

Note: On Fedora you can also use OpenVPN to implement a virtual private network (VPN). OpenVPN uses the Web's SSL secure connections (Secure Socket Layer) and a server to manage connections. Set up and management is very easy. Check **www.openvpn.org** for more details.

IPsec Protocols

IPsec is made up of several protocols that provide authentication (AH), encryption (ESP), and the secure exchange of encryption keys (IKE). The Authentication Header protocol (AH) confirms that the packet was sent by the sender, and not by someone else. IPsec also includes an integrity check to detect any tampering in transit. Packets are encrypted using the Encapsulating Security Payload (ESP). Encryption and decryption are performed using secret keys shared by the sender and the receiver. These keys are themselves transmitted using the Internet Key Exchange protocol, which provides a secure exchange. ESP encryption can degrade certain compression

transmission methods, such as PPP for dial-up Internet connections. To accommodate these compression methods, IPsec provides the IP Payload Compression Protocol (IPComp), with which packets can be compressed before being sent.

Encrypted authentication and integrity checks are included using Hash Methods Authentication Codes (HMAC) generated from hash security methods like SHA2 using a secret key. The HMAC is the included in the IPsec header, which the receiver can then check with the secret key. Encryption of transmitted data is performed by symmetric encryption methods like 3DES, Blowfish, and DES.

The AH, ESP, and IPComp protocols are incorporated into the Linux kernel. The IKE protocol is implemented as a separate daemon. It simply provides a way to share secret keys, and can be replaced by other sharing methods.

Note: You can now quickly and effectively configure IPsec connections using Network Manager.

IPsec Modes

You can use IPsec capabilities for either normal transport or packet tunneling. With normal transport, packets are encrypted and sent to the next destination. The normal transport mode is used to implement direct host-to-host encryption, where each host handles the IPsec encryption process. Packet tunneling is used to encrypt transmissions between gateways, letting the gateways handle the IPsec encryption process for traffic directed to or from an entire network, rather than having to configure IPsec encryption for each host. With packet tunneling, the packets are encapsulated with new headers for a specific destination, enabling you to implement virtual private networks (VPNs). Packets are directed to VPN gateways, which encrypt and send on local network packets.

Note: You can choose to encrypt packets for certain hosts or for those passing through specific ports.

IPsec Security Databases

The packets you choose to encrypt are designated by the IPsec Security Policy Database (SPD). The method you use to encrypt them is determined by the IPsec Security Association Database (SAD). The SAD associates an encryption method and key with a particular connection or kind of connection. The connections to be encrypted are designated in the Security Policy Database.

IPsec Tools

Several IPsec tools are provided with which you can manage your IPsec connections. These are included in the Red Hat Fedora **ipsec-tools** RPM package. With **`setkey`**, you can manage both the policy and association databases. The **`racoon`** tool configures the key exchange process to implement secure decryption key exchanges across connections. To administer your IPsec connections you can use **`racoonctl`**. For example, the **`show-sa`** option will display your security associations and the **`vpn-connect`** will establish a VPN connection.

Configuring IPsec with system-config-network

The system-config-network tool now provides support for implementing IPsec connections. On the system-config-network tool, select the IPsec panel and click New to start the IPsec settings wizard for creating an IPsec connection. You are first asked to enter a nickname for the connection and to specify if you want it started automatically. You then choose the connection type. This can be either a direct host-to-host connection or a connection between two networks. A network connection implements a virtual private network (VPN) and runs IPsec in tunnel mode. (Both the host and VPN connections are described in detail in the following sections.) You then select the kind of encryption you want to use. This can either be manual or use IKE, letting **racoon** automatically manage the encryption and authentication process.

You then will configure both your local and remote connections, starting with the local settings. For a host-to-host connection, you need only enter the IP address for the remote host. For a VPN, you will have to enter corresponding addresses for the local and remote networks. For the local network, you will need to enter the IP addresses for the local network, the local network's gateway computer, and the local network's netmask. For the remote VPN connection, you will need the remote IP address, the remote network's address, its netmask, and its gateway address. Finally, you enter the authentication key. Click the Generate button to create one.

A final screen will display your entries. Click Apply to save them. Your connection will appear in the IPsec panel, showing its type, destination, and nickname. To establish a connection, select the IPsec connection and click Activate. This will run the **ifup-ipsec** script in the **/etc/sysconfig/network-scripts** directory, which will execute IPsec tools such as **setkey** and **racoon** to establish your connection. Configuration data will be kept in the /**etc/sysconfig/networking/devices** directory, using the name of the IPsec connections. For example, configuration information on the **myipsec** IPsec connection is kept in the **ifcfg-myipsec** file. Corresponding keys for each connection are kept in the keys files, including **keys-myipsec**. A sample **ifcfg** configuration file for a VPN is shown here. The IKE method is a private shared key (PSK). The destination (remote) gateway is 10.0.0.1, and the source (local) gateway is 192.168.0.1. The destination (remote) network address is 10.0.0.0/24, and the source (local) address is 192.168.0.0/24. The destination host is 10.0.0.2.

```
ONBOOT=no
IKE_METHOD=PSK
DSTGW=10.0.0.1
SRCGW=192.168.0.1
DSTNET=10.0.0.0/24
SRCNET=192.168.0.0/24
DST=10.0.0.2
TYPE=IPSEC
```

The corresponding keys file would specify the key used.

```
IKE_PSK=myvpnkey
```

To configure your IPsec connections, you can use the **setkey** tool. This tool contains several instructions for managing rules in the IPsec policy and security databases. You use the **add** instruction to add a security association to the security database (SAD), and the **spdadd** instruction to add a policy to the policy database (SPD)

.

fedora

14. Firewalls

system-config-firewall
Firewalls: IPtables, NAT, and ip6tables
Packet Filtering
Network Address Translation (NAT)
Packet Mangling: the Mangle Table
IPtables Scripts
IP Masquerading

Most systems currently connected to the Internet are open to attempts by outside users to gain unauthorized access. Outside users can try to gain access directly by setting up an illegal connection, by intercepting valid communications from users remotely connected to the system, or by pretending to be a valid user. Firewalls, encryption, and authentication procedures are ways of protecting against such attacks. A *firewall* prevents any direct unauthorized attempts at access, *encryption* protects transmissions from authorized remote users, and *authentication* verifies that a user requesting access has the right to do so. The current Linux kernel incorporates support for firewalls using the Netfilter (IPtables) packet filtering package (the previous version, IP Chains, is used on older kernel versions). To implement a firewall, you simply provide a series of rules to govern what kind of access you want to allow on your system. If that system is also a gateway for a private network, the system's firewall capability can effectively help protect the network from outside attacks.

Web Site	Security Application
www.netfilter.org	Netfilter project, Iptables, and NAT
www.netfilter.org/ipchains	IP Chains firewall
www.openssh.org	Secure Shell encryption
www.squid-cache.org	Squid Web Proxy server
http://web.mit.edu/Kerberos	Kerberos network authentication

Table 14-1: Network Security Applications

To provide protection for remote communications, transmission can be simply encrypted. For Linux systems, you can use the Secure Shell (SSH) suite of programs to encrypt any transmissions, preventing them from being read by anyone else. Kerberos authentication provides another level of security whereby individual services can be protected, allowing use of a service only to users who are cleared for access. Outside users may also try to gain unauthorized access through any Internet services you may be hosting, such as a Web site. In such a case, you can set up a proxy to protect your site from attack. For Linux systems, use Squid proxy software to set up a proxy to protect your Web server. Table 14-1 lists several network security applications commonly used on Linux.

system-config-firewall

Fedora 10 uses a new system configuration tool called system-config-firewall, to set up your firewall (see Figure 14-1) (SELinux is configured with a separate tool called system-config-selinux). To run system-config-firewall, select System | Adminsitration | Firewall.

You can run your firewall on a stand-alone system directly connected to the Internet, or on a gateway system that connects a local network to the Internet. Most networks now use dedicated routers for Internet access which have their own firewalls. If, instead, you decide to use a Linux system as a gateway, it will have at least two network connections, one for the local network and an Internet connection device for the Internet. Make sure that the firewall is applied to the Internet device, not to your local network.

The top button bar has buttons for a Firewall wizard, Apply button to effect any changes, Reload to restore your saved firewall, and Disable and Enable buttons. You can enable or disable

your firewall with the Enable and Disable buttons (see Figure 14-2). If the Firewall is active, only the Disable button can be used, and visa versa.

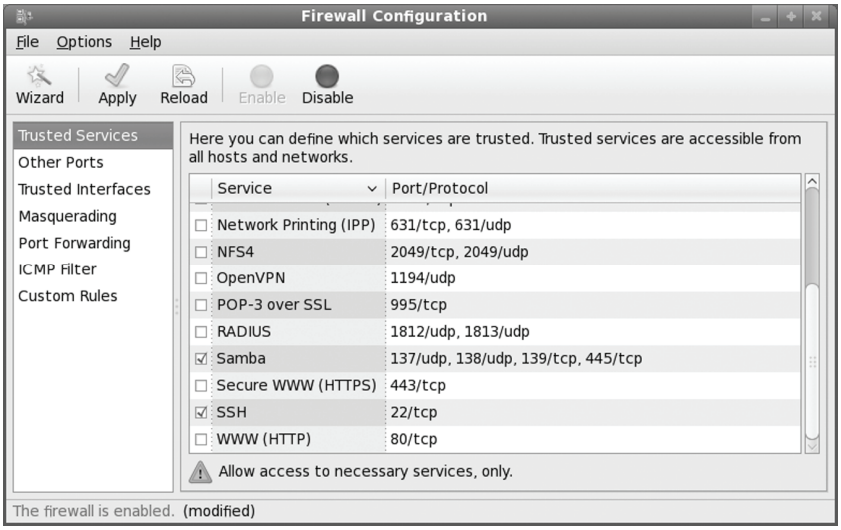


Figure 14-1: The system-config-firewall showing Firewall Trusted Services

A default configuration will already be in place. To create your own basic custom configuration just run the Wizard, by clicking the Wizard button.

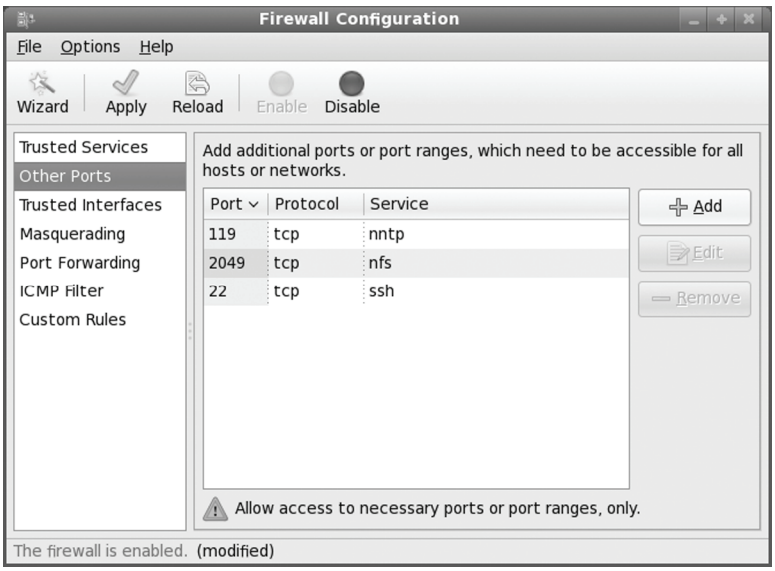


Figure 14-2: Firewall Other Ports configuration

You can also make changes directly to different firewall components. The Firewall Configuration window will show several panes, selected from entries on a sidebar. The panes are Trusted Services, Other Ports, Trusted Interfaces, Masquerading, and Custom Rules.

The feature many users may want to change are the Trusted services. A Linux system will often be used to run servers for a network. If you are creating a strong firewall but still want to run a service such as a Web server, allow users to perform FTP file transfers on the Internet, permit Samba desktop browsing, or allow remote encrypted connections such as SSH, you will have to specify them in the Trusted Services panel. Samba desktop browsing lets you access your Samba shares, like remote Windows file systems, from your GNOME or KDE desktops.

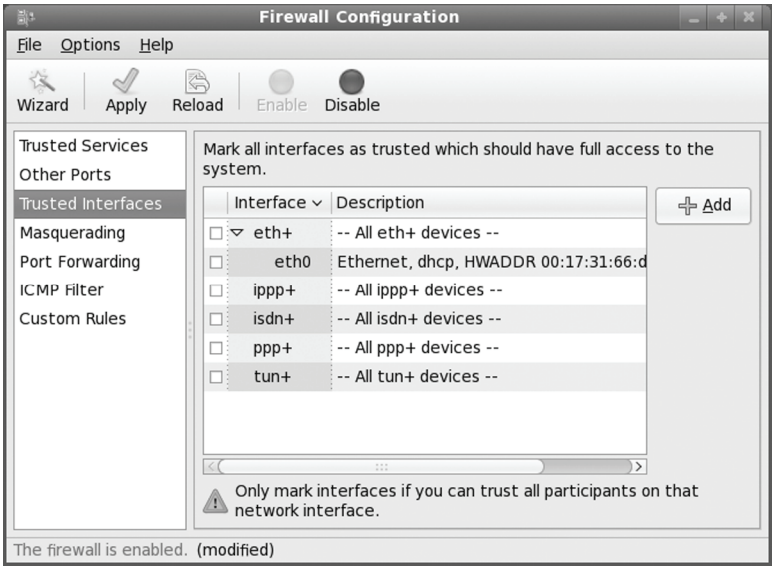


Figure 14-3: Firewall Trusted Interfaces

The Other Ports section lets you specify ports that you may want opened for certain services, like BitTorrent (see Figure 14-3). Click the Add button to open a dialog where you can enter the port number along with the protocol to control (tcp or udp).

The Trusted interfaces panel lets you select those hardware network connections like NIC cards, modems, or wireless connections that can be trusted (see Figure 14-4). This is usually applied to interfaces connected to a local network or directly to a local router or server (like a print server). Interfaces connected to wider network like the Internet should not be trusted.

If your system is being used as gateway to the Internet for your local network, you can implement masquerading to hide your local hosts from outside access from the Internet (see Figure 14-5). This, though, also requires IP forwarding which is automatically enabled when you choose masquerading. Local hosts will still be able to access the Internet, but they will masquerade as your gateway system. You would select for masquerading the interface that is connected to the Internet. Masquerading is available only for IPv4 networks, not IPv6 networks.

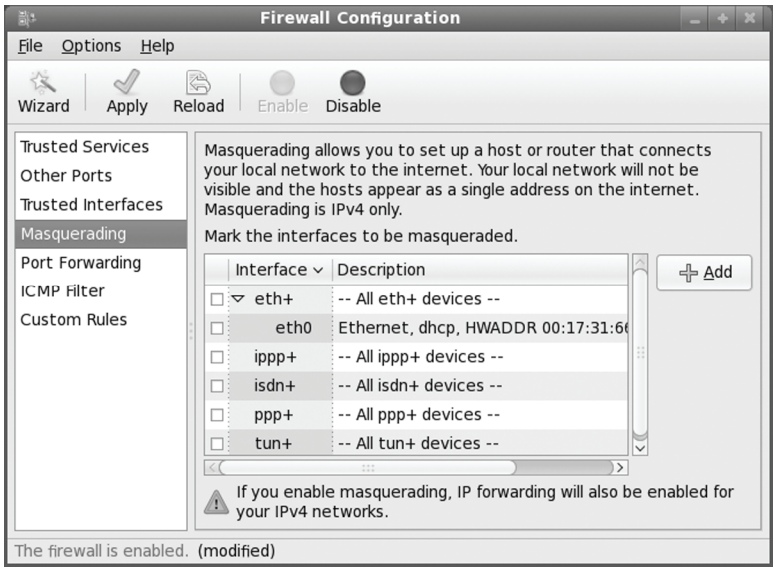


Figure 14-4: Firewall Masquerading

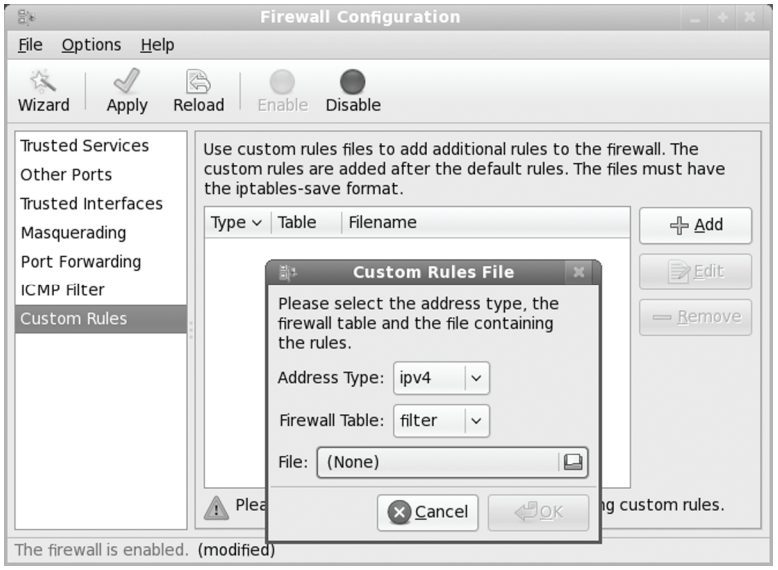


Figure 14-5: Firewall Custom Rules

The Custom Rules panel lets you load an IP-tables file that can hold customized firewall rules (see Figure 14-6). Instead of having to choose whether to use system-config-firewall or to create an entire set of rules in an IP-tables file, you can use system-config-firewall to automatically configure the standard firewall rules and then set up an IP-tables file that holds only customized

rules for your system. These will be added to the standard set implemented by system-config-firewall. When you load the file you will also have to determine whether the IPtables fie holds IPv4 or IPv6 protocol rules (Address type). You will also have to select which table the firewall rules are meant for. There are three choices: filter (Netfilter rules), mangle (the mangle table used for rewriting addresses), and the nat table. Use the File entry to open a browser to locate the IPtables file you want to load.

Firewalls: IPtables, NAT, and ip6tables

A good foundation for your network's security is to set up a Linux system to operate as a firewall for your network, protecting it from unauthorized access. You can use a firewall to implement either packet filtering or proxies. *Packet filtering* is simply the process of deciding whether a packet received by the firewall host should be passed on into the local network. The packet-filtering software checks the source and destination addresses of the packet and sends the packet on, if it's allowed. Even if your system is not part of a network but connects directly to the Internet, you can still use the firewall feature to control access to your system. Of course, this also provides you with much more security.

With proxies, you can control access to specific services, such as Web or FTP servers. You need a proxy for each service you want to control. The Web server has its own Web proxy, while an FTP server has an FTP proxy. Proxies can also be used to cache commonly used data, such as Web pages, so that users needn't constantly access the originating site. The proxy software commonly used on Linux systems is Squid.

An additional task performed by firewalls is network address translation (NAT). Network address translation redirects packets to appropriate destinations. It performs tasks such as redirecting packets to certain hosts, forwarding packets to other networks, and changing the host source of packets to implement IP masquerading.

Note: The IP Chains package is the precursor to IPtables that was used on Linux systems running the 2.2 kernel. It is still in use on some Linux systems. The Linux Web site for IP Chains, which is the successor to ipfwadm used on older versions of Linux, is currently **www.netfilter.org/ipchains**. IP Chains is no longer included with many Linux distributions.

The Netfilter software package implements both packet filtering and NAT tasks for the Linux 2.4 kernel and above. The Netfilter software is developed by the Netfilter Project, which you can find out more about at **www.netfilter.org**.

IPtables

The command used to execute packet filtering and NAT tasks is **`iptables`**, and the software is commonly referred to as simply IPtables. However, Netfilter implements packet filtering and NAT tasks separately using different tables and commands. A table will hold the set of commands for its application. This approach streamlines the packet-filtering task, letting IPtables perform packet-filtering checks without the overhead of also having to do address translations. NAT operations are also freed from being mixed in with packet-filtering checks.

You use the iptables command for packet filtering, NAT tasks, and packet mangling. Each operation has its own table of rules: filter for packet filtering, nat for NAT tasks, and mangle for packet mangling. For NAT you specify the NAT table with the **-t nat** option. For the mangle table

you use the **-t mangle** option. The packet filtering is the default. It can be specified with the **-t filte**r **-t filter** option, but it's usually left out, assuming that if a table is not specified it is a filter operation. In addition, netfilter also handles certain exemptions to connection tracking operations in a raw table.

ip6tables

The ip6tables package provides support for IPv6 addressing. It is identical to IPtables except that it allows the use of IPv6 addresses instead of IPv4 addresses. Both filter and mangle tables are supported in ip6tables, but not NAT tables. The filter tables support the same options and commands as in IPtables. The mangle tables will allow specialized packet changes like those for IPtables, using PREROUTING, INPUT, OUTPUT, FORWARD, and POSTROUTING rules. Some extensions have ipv6 labels for their names, such as ipv6-icmp, which corresponds to the IPtables icmp extension. The ipv6headers extension is used to select IPv6 headers.

Modules

Unlike its predecessor, IP Chains, Netfilter is designed to be modularized and extensible. Capabilities can be added in the form of modules such as the state module, which adds connection tracking. Most modules are loaded as part of the IPtables service. Others are optional; you can elect to load them before installing rules. The IPtables modules are located at **/usr/lib/***kernel-version***/kernel/net/ipv4/netfilter**, where *kernel-version* is your kernel number. For IPv6 modules, check the **ipv6/netfilter** directory. Modules that load automatically will have an **ipt_** prefix, and optional ones have just an **ip_** prefix. If you are writing you own iptables script, you would have to add **`modprobe`** commands to load optional modules directly.

Optional modules can be specified as a list assigned to the IPTABLES_MODULES parameters located in the **/etc/sysconfig/iptables-config** file, which is used by the iptables service script. Whenever you start IPtables with the iptables service script, they will be automatically loaded. For IPv6 configuration, you use the **ip6tables-config** file.

```
service iptables start
```

If you are writing you own iptables script, instead of using the Red Hat or Fedora service iptables script, you would have to add **modprobe** commands to load optional modules directly

Packet Filtering

Netfilter is essentially a framework for packet management that can check packets for particular network protocols and notify parts of the kernel listening for them. Built on the Netfilter framework is the packet selection system implemented by IPtables. With IPtables, different tables of rules can be set up to select packets according to differing criteria. Netfilter currently supports three tables: filter, nat, and mangle. Packet filtering is implemented using a filter table that holds rules for dropping or accepting packets. Network address translation operations such as IP masquerading are implemented using the NAT table that holds IP masquerading rules. The mangle table is used for specialized packet changes. Changes can be made to packets before they are sent out, when they are received, or as they are being forwarded. This structure is extensible in that new modules can define their own tables with their own rules. It also greatly improves efficiency. Instead of all packets checking one large table, they access only the table of rules they need to.

IP table rules are managed using the **iptables** command. For this command, you will need to specify the table you want to manage. The default is the filter table, which need not be specified. You can list the rules you have added at any time with the **-L** and **-n** options, as shown here. The **-n** option says to use only numeric output for both IP addresses and ports, avoiding a DNS lookup for hostnames. You could, however, just use the **-L** option to see the port labels and hostnames:

```
iptables -L -n
```

Chains

Rules are combined into different chains. The kernel uses chains to manage packets it receives and sends out. A *chain* is simply a checklist of rules. These rules specify what action to take for packets containing certain headers. The rules operate with an if-then-else structure. If a packet does not match the first rule, the next rule is then checked, and so on. If the packet does not match any rules, the kernel consults chain policy. Usually, at this point the packet is rejected. If the packet does match a rule, it is passed to its target, which determines what to do with the packet. The standard targets are listed in Table 14-2. If a packet does not match any of the rules, it is passed to the chain's default target.

Target	Function
ACCEPT	Allow packet to pass through the firewall.
DROP	Deny access by the packet.
REJECT	Deny access and notify the sender.
QUEUE	Send packets to user space.
RETURN	Jump to the end of the chain and let the default target process it.

Table 14-2: IPtables Targets

Targets

A *target* could, in turn, be another chain of rules, even a chain of user-defined rules. A packet could be passed through several chains before finally reaching a target. In the case of user-defined chains, the default target is always the next rule in the chains from which it was called. This sets up a procedure- or function call–like flow of control found in programming languages. When a rule has a user-defined chain as its target, when activated, that user-defined chain is executed. If no rules are matched, execution returns to the next rule in the originating chain.

Tip: Specialized targets and options can be added by means of kernel patches provided by the Netfilter site. For example, the SAME patch returns the same address for all connections. A patch-o-matic option for the Netfilter make file will patch your kernel source code, adding support for the new target and options. You can then rebuild and install your kernel.

Firewall and NAT Chains

The kernel uses three firewall chains: INPUT, OUTPUT, and FORWARD. When a packet is received through an interface, the INPUT chain is used to determine what to do with it. The kernel then uses its routing information to decide where to send it. If the kernel sends the packet to

another host, the FORWARD chain is checked. Before the packet is actually sent, the OUTPUT chain is also checked. In addition, two NAT table chains, POSTROUTING and PREROUTING, are implemented to handle masquerading and packet address modifications. The built-in Netfilter chains are listed in Table 14-3.

Chain	Description
INPUT	Rules for incoming packets
OUTPUT	Rules for outgoing packets
FORWARD	Rules for forwarded packets
PREROUTING	Rules for redirecting or modifying incoming packets, NAT table only
POSTROUTING	Rules for redirecting or modifying outgoing packets, NAT table only

Table 14-3: Netfilter Built-in Chains

Option	Function
-A *chain*	Appends a rule to a chain.
-D *chain* **[** *rulenum***]**	Deletes matching rules from a chain. Deletes rule *rulenum* (1 = first) from *chain.*
-I *chain* **[***rulenum***]**	Inserts in *chain* as *rulenum* (default 1 = first).
-R *chain rulenum*	Replaces rule *rulenum* (1 = first) in *chain.*
-L [*chain***]**	Lists the rules in *chain* or all chains.
-E [*chain***]**	Renames a chain.
-F [*chain***]**	Deletes (flushes) all rules in *chain* or all chains.
-R *chain*	Replaces a rule; rules are numbered from 1.
-Z [*chain***]**	Zero counters in *chain* or all chains.
-N *chain*	Creates a new user-defined chain.
-X *chain*	Deletes a user-defined chain.
-P *chain target*	Changes policy on *chain* to *target.*

Table 14-4: IPtables Commands

Adding and Changing Rules

You add and modify chain rules using the **iptables** commands. An **iptables** command consists of the command **iptables**, followed by an argument denoting the command to execute. For example, **iptables -A** is the command to add a new rule, whereas **iptables -D** is the command to delete a rule. The **iptables** commands are listed in Table 14-4. The following

command simply lists the chains along with their rules currently defined for your system. The output shows the default values created by **iptables** commands.

```
iptables -L -n
Chain input (policy ACCEPT):
Chain forward (policy ACCEPT):
Chain output (policy ACCEPT):
```

To add a new rule to a chain, you use **-A**. Use **-D** to remove it, and **-R** to replace it. Following the command, list the chain to which the rule applies, such as the INPUT, OUTPUT, or FORWARD chain, or a user-defined chain. Next, you list different options that specify the actions you want taken (most are the same as those used for IP Chains, with a few exceptions). The **-s** option specifies the source address attached to the packet, **-d** specifies the destination address, and the **-j** option specifies the target of the rule. The ACCEPT target will allow a packet to pass. The **-i** option now indicates the input device and can be used only with the INPUT and FORWARD chains. The **-o** option indicates the output device and can be used only for OUTPUT and FORWARD chains. Table 14-5 lists several basic options.

IPtables Options

The IPtables package is designed to be extensible, and there are number of options with selection criteria that can be included with IPtables. For example, the TCP extension includes the **--syn** option that checks for SYN packets. The ICMP extension provides the **--icmp-type** option for specifying ICMP packets as those used in ping operations. The limit extension includes the **--limit** option, with which you can limit the maximum number of matching packets in a specified time period, such as a second.

Note: In IPtables commands, chain names have to be entered in uppercase, as with the chain names INPUT, OUTPUT, and FORWARD.

In the following example, the user adds a rule to the INPUT chain to accept all packets originating from the address 192.168.0.55. Any packets that are received (**INPUT**) whose source address (**-s**) matches 192.168.0.55 are accepted and passed through (**-j ACCEPT**):

```
iptables -A INPUT -s 192.168.0.55 -j ACCEPT
```

Accepting and Denying Packets: DROP and ACCEPT

There are two built-in targets, DROP and ACCEPT. Other targets can be either user-defined chains or extensions added on, such as REJECT. Two special targets are used to manage chains, RETURN and QUEUE. RETURN indicates the end of a chain and returns to the chain it started from. QUEUE is used to send packets to user space.

```
iptables -A INPUT -s www.myjunk.com -j DROP
```

You can turn a rule into its inverse with an **!** symbol. For example, to accept all incoming packets except those from a specific address, place an **!** symbol before the **-s** option and that address. The following example will accept all packets except those from the IP address 192.168.0.45:

```
iptables -A INPUT -j ACCEPT ! -s 192.168.0.45
```

You can specify an individual address using its domain name or its IP number. For a range of addresses, you can use the IP number of their network and the network IP mask. The IP mask can be an IP number or simply the number of bits making up the mask. For example, all of the addresses in network 192.168.0 can be represented by 192.168.0.0/225.255.255.0 or by 192.168.0.0/24. To specify any address, you can use 0.0.0.0/0.0.0.0 or simply 0/0. By default, rules reference any address if no **-s** or **-d** specification exists. The following example accepts messages coming in that are from (source) any host in the 192.168.0.0 network and that are going (destination) anywhere at all (the **-d** option is left out or could be written as **-d 0/0**):

```
iptables -A INPUT -s 192.168.0.0/24  -j ACCEPT
```

The IPtables rules are usually applied to a specific network interface such as the Ethernet interface used to connect to the Internet. For a single system connected to the Internet, you will have two interfaces, one that is your Internet connection and a loopback interface (**lo**) for internal connections between users on your system. The network interface for the Internet is referenced using the device name for the interface. For example, an Ethernet card with the device name **/dev/eth0** would be referenced by the name **eth0**. A modem using PPP protocols with the device name **/dev/ppp0** would have the name **ppp0**. In IPtables rules, you use the **-i** option to indicate the input device; it can be used only with the INPUT and FORWARD chains. The **-o** option indicates the output device and can be used only for OUTPUT and FORWARD chains. Rules can then be applied to packets arriving and leaving on particular network devices. In the following examples, the first rule references the Ethernet device **eth0**, and the second references the localhost:

```
iptables -A INPUT -j DROP -i eth0 -s 192.168.0.45
iptables -A INPUT -j ACCEPT  -i lo
```

User-Defined Chains

With IPtables, the FORWARD and INPUT chains are evaluated separately. One does not feed into the other. This means that if you want to completely block certain addresses from passing through your system, you will need to add both a FORWARD rule and an INPUT rule for them.

```
iptables -A INPUT -j DROP -i eth0 -s 192.168.0.45
iptables -A FORWARD -j DROP -i eth0 -s 192.168.0.45
```

A common method for reducing repeated INPUT and FORWARD rules is to create a user chain that both the INPUT and FORWARD chains feed into. You define a user chain with the **-N** option. The next example shows the basic format for this arrangement. A new chain is created called incoming (it can be any name you choose). The rules you would define for your FORWARD and INPUT chains are now defined for the incoming chain. The INPUT and FORWARD chains then use the incoming chain as a target, jumping directly to it and using its rules to process any packets they receive.

```
iptables -N incoming

iptables -A incoming -j DROP -i eth0 -s 192.168.0.45
iptables -A incoming -j ACCEPT  -i lo

iptables -A FORWARD -j incoming
iptables -A INPUT -j incoming
```

Option	Function
-p [!] *proto*	Specifies a protocol, such as TCP, UDP, ICMP, or ALL.
-s [!] *address*[/*mask*] **[!] [***port*[:*port*]]	Source address to match. With the *port* argument, you can specify the port.
--sport [!] [*port*[:*port*]]	Source port specification. You can specify a range of ports using the colon, *port*:*port.*
-d [!] *address*[/*mask*] **[!] [***port*[:*port*]]	Destination address to match. With the *port* argument, you can specify the port.
--dport [!][*port*[:*port*]]	Destination port specification.
--icmp-type [!] *typename*	Specifies ICMP type.
-i [!] *name*[+]	Specifies an input network interface using its name (for example, **eth0**). The **+** symbol functions as a wildcard. The **+** attached to the end of the name matches all interfaces with that prefix (**eth+** matches all Ethernet interfaces). Can be used only with the INPUT chain.
-j *target* [*port*]	Specifies the target for a rule (specify [*port*] for REDIRECT target).
--to-source < *ipaddr*> **[-<** *ipaddr*>][: *port*- *port*]	Used with the SNAT target, rewrites packets with new source IP address.
--to-destination < *ipaddr*> **[-<** *ipaddr*>][: *port*- *port*]	Used with the DNAT target, rewrites packets with new destination IP address.
-n	Numeric output of addresses and ports, used with **-L**.
-o [!] *name*[+]	Specifies an output network interface using its name (for example, **eth0**). Can be used only with FORWARD and OUTPUT chains.
-t *table*	Specifies a table to use, as in **-t nat** for the NAT table.
-v	Verbose mode, shows rule details, used with **-L**.
-x	Expands numbers (displays exact values), used with **-L**.
[!] -f	Matches second through last fragments of a fragmented packet.
[!] -V	Prints package version.
!	Negates an option or address.

-m	Specifies a module to use, such as state.
--state	Specifies options for the state module such as NEW, INVALID, RELATED, and ESTABLISHED. Used to detect packet's state. NEW references SYN packets (new connections).
--syn	SYN packets, new connections.
--tcp-flags	TCP flags: SYN, ACK, FIN, RST, URG, PS, and ALL for all flags.
--limit	Option for the limit module (**-m limit**). Used to control the rate of matches, matching a given number of times per second.
--limit-burst	Option for the limit module (**-m limit**). Specifies maximum burst before the limit kicks in. Used to control denial-of-service attacks.

Table 14-5: IPtables Options

ICMP Packets

Firewalls often block certain Internet Control Message Protocol (ICMP) messages. ICMP redirect messages, in particular, can take control of your routing tasks. You need to enable some ICMP messages, however, such as those needed for ping, traceroute, and particularly destination-unreachable operations. In most cases, you always need to make sure destination-unreachable packets are allowed; otherwise, domain name queries could hang. Some of the more common ICMP packet types are listed in Table 14-6. You can enable an ICMP type of packet with the **--icmp-type** option, which takes as its argument a number or a name representing the message. The following examples enable the use of echo-reply, echo-request, and destination-unreachable messages, which have the numbers 0, 8, and 3:

```
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 --icmp -type  echo-reply -d 10.0.0.1
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 --icmp-type  echo-request -d
10.0.0.1
iptables -A INPUT -j ACCEPT -p icmp -i eth0 --icmp-type  destination-unreachable
-d
10.0.0.1
```

Their rule listing will look like this:

```
ACCEPT     icmp --  0.0.0.0/0            10.0.0.1           icmp type 0
ACCEPT     icmp --  0.0.0.0/0            10.0.0.1           icmp type 8
ACCEPT     icmp --  0.0.0.0/0            10.0.0.1           icmp type 3
```

Ping operations need to be further controlled to avoid the ping-of-death security threat. You can do this several ways. One way is to deny any ping fragments. Ping packets are normally very small. You can block ping-of-death attacks by denying any ICMP packet that is a fragment. Use the **-f** option to indicate fragments.

```
iptables -A INPUT -p icmp -j DROP -f
```

Another way is to limit the number of matches received for ping packets. You use the limit module to control the number of matches on the ICMP ping operation. Use **-m limit** to use

the limit module, and **--limit** to specify the number of allowed matches. **1/s** will allow one match per second.

```
iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s -j
ACCEPT
```

Number	Name	Required By
0	echo-reply	ping
3	destination-unreachable	Any TCP/UDP traffic
5	redirect	Routing if not running routing daemon
8	echo-request	ping
11	time-exceeded	traceroute

Table 14-6: Common ICMP Packets

Controlling Port Access

If your system is hosting an Internet service, such as a Web or FTP server, you can use IPtables to control access to it. You can specify a particular service by using the source port (**--sport**) or destination port **(--dport**) options with the port that the service uses. IPtables lets you use names for ports such as **www** for the Web server port. The names of services and the ports they use are listed in the **/etc/services** file, which maps ports to particular services. For a domain name server, the port would be **domain**. You can also use the port number if you want, preceding the number with a colon. The following example accepts all messages to the Web server located at 192.168.0.43:

```
iptables -A INPUT -d 192.168.0.43 --dport www -j ACCEPT
```

You can also use port references to protect certain services and deny others. This approach is often used if you are designing a firewall that is much more open to the Internet, letting users make freer use of Internet connections. Certain services you know can be harmful, such as Telnet and NTP, can be denied selectively. For example, to deny any kind of Telnet operation on your firewall, you can drop all packets coming in on the Telnet port, 23. To protect NFS operations, you can deny access to the port used for the portmapper, 111. You can use either the port number or the port name.

```
# deny outside access to portmapper port on firewall.
iptables -A arriving  -j DROP -p tcp -i eth0  --dport 111
# deny outside access to telnet port on firewall.
iptables -A arriving  -j DROP -p tcp -i eth0  --dport telnet
```

The rule listing will look like this:

```
DROP      tcp  --  0.0.0.0/0    0.0.0.0/0     tcp dpt:111
DROP      tcp  --  0.0.0.0/0    0.0.0.0/0     tcp dpt:23
```

One port-related security problem is access to your X server ports that range from 6000 to 6009. On a relatively open firewall, these ports could be used to illegally access your system through your X server. A range of ports can be specified with a colon, as in 6000:6009. You can

also use x11 for the first port, x11:6009. Sessions on the X server can be secured by using SSH, which normally accesses the X server on port 6010.

```
iptables -A arriving  -j DROP -p tcp -i eth0  --dport 6000:6009
```

Common ports checked and their labels are shown here:

Service	Port Number	Port Label
Auth	113	auth
Finger	79	finger
FTP	21	ftp
NTP	123	ntp
Portmapper	111	sunrpc
Telnet	23	telnet
Web server	80	www

Packet States: Connection Tracking

One of the more useful extensions is the state extension, which can easily detect tracking information for a packet. Connection tracking maintains information about a connection such as its source, destination, and port. It provides an effective means for determining which packets belong to an established or related connection. To use connection tracking, you specify the state module first with **`-m state`**. Then you can use the **`--state`** option. Here you can specify any of the following states:

State	Description
NEW	A packet that creates a new connection
ESTABLISHED	A packet that belongs to an existing connection
RELATED	A packet that is related to, but not part of, an existing connection, such as an ICMP error or a packet establishing an FTP data connection
INVALID	A packet that could not be identified for some reason
RELATED+REPLY	A packet that is related to an established connection, but not part of one directly

If you are designing a firewall that is meant to protect your local network from any attempts to penetrate it from an outside network, you may want to restrict packets coming in. Simply denying access by all packets is unfeasible because users connected to outside servers—say, on the Internet—must receive information from them. You can, instead, deny access by a particular kind of packet used to initiate a connection. The idea is that an attacker must initiate a connection from the outside. The headers of these kinds of packets have their SYN bit set on and their FIN and ACK bits empty. The state module's NEW state matches on any such SYN packet. By specifying a DROP target for such packets, you deny access by any packet that is part of an

attempt to make a connection with your system. Anyone trying to connect to your system from the outside is unable to do so. Users on your local system who have initiated connections with outside hosts can still communicate with them. The following example will drop any packets trying to create a new connection on the **eth0** interface, though they will be accepted on any other interface:

```
iptables -A INPUT -m state --state NEW -i eth0 -j DROP
```

You can use the **!** operator on the **eth0** device combined with an ACCEPT target to compose a rule that will accept any new packets except those on the **eth0** device. If the **eth0** device is the only one that connects to the Internet, this still effectively blocks outside access. At the same time, input operation for other devices such as your localhost are free to make new connections. This kind of conditional INPUT rule is used to allow access overall with exceptions. It usually assumes that a later rule such as a chain policy will drop remaining packets.

```
iptables -A INPUT -m state --state NEW ! -i eth0 -j ACCEPT
```

The next example will accept any packets that are part of an established connection or related to such a connection on the **eth0** interface:

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Tip: You can use the iptstate tool to display the current state table.

Specialized Connection Tracking: ftp, irc, Amanda, tftp.

To track certain kinds of packets, IPtables uses specialized connection tracking modules. These are optional modules that you have to have loaded manually. To track passive FTP connections, you would have to load the ip_conntrack_ftp module. To add NAT table support, you would also load the ip_nat_ftp module. For IRC connections, you use ip_conntrack_irc and ip_nat_irc. There are corresponding modules for Amanda (the backup server) and TFTP (Trivial FTP).

If you are writing your own iptables script, you would have to add **modprobe** commands to load the modules.

```
modprobe ip_conntrack ip_conntrack_ftp ip_nat_ftp
modprobe ip_conntrack_amanda ip_nat_amanda
```

Network Address Translation (NAT)

Network address translation (NAT) is the process whereby a system will change the destination or source of packets as they pass through the system. A packet will traverse several linked systems on a network before it reaches its final destination. Normally, they will simply pass the packet on. However, if one of these systems performs a NAT on a packet, it can change the source or destination. A packet sent to a particular destination could have its destination address changed. To make this work, the system also needs to remember such changes so that the source and destination for any reply packets are altered back to the original addresses of the packet being replied to.

NAT is often used to provide access to systems that may be connected to the Internet through only one IP address. Such is the case with networking features such as IP masquerading,

support for multiple servers, and transparent proxying. With IP masquerading, NAT operations will change the destination and source of a packet moving through a firewall/gateway linking the Internet to computers on a local network. The gateway has a single IP address that the other local computers can use through NAT operations. If you have multiple servers but only one IP address, you can use NAT operations to send packets to the alternate servers. You can also use NAT operations to have your IP address reference a particular server application such as a Web server (transparent proxy). NAT tables are not implemented for ip6tables.

Note: Using proxies, you can control access to specific services, such as web or FTP servers. You need a proxy for each service you want to control. The web server has its own web proxy, while an FTP server has an FTP proxy. Proxies can also be used to cache commonly used data, such as web pages, so that users needn't constantly access the originating site. The proxy software commonly used on Linux systems is Squid.

Adding NAT Rules

Packet selection rules for NAT operations are added to the NAT table managed by the **iptables** command. To add rules to the NAT table, you have to specify the NAT table with the **-t** option. Thus to add a rule to the NAT table, you would have to specify the NAT table with the **-t nat** option as shown here:

```
iptables -t nat
```

With the **-L** option, you can list the rules you have added to the NAT table:

```
iptables -t nat -L -n
```

Adding the **-n** option will list IP addresses and ports in numeric form. This will speed up the listing, as iptables will not attempt to do a DNS lookup to determine the hostname for the IP address.

Nat Targets and Chains

In addition, there are two types of NAT operations: source NAT, specified as SNAT target, and destination NAT, specified as DNAT target. SNAT target is used for rules that alter source addresses, and DNAT target, for those that alter destination addresses.

Three chains in the NAT table are used by the kernel for NAT operations. These are PREROUTING, POSTROUTING, and OUTPUT. PREROUTING is used for destination NAT (DNAT) rules. These are packets that are arriving. POSTROUTING is used for source NAT (SNAT) rules. These are for packets leaving. OUTPUT is used for destination NAT rules for locally generated packets.

As with packet filtering, you can specify source (**-s**) and destination (**-d**) addresses, as well as the input (**-i**) and output (**-o**) devices. The **-j** option will specify a target such as MASQUERADE. You would implement IP masquerading by adding a MASQUERADE rule to the POSTROUTING chain:

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

To change the source address of a packet leaving your system, you would use the POSTROUTING rule with the SNAT target. For the SNAT target, you use the **--to-source** option to specify the source address:

```
# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 192.168.0.4
```

To change the destination address of packets arriving on your system, you would use the PREROUTING rule with the DNAT target and the **--to-destination** option:

```
# iptables -t nat -A PRETROUTING -i eth0 \
             -j DNAT --to-destination 192.168.0.3
```

Specifying a port lets you change destinations for packets arriving on a particular port. In effect, this lets you implement port forwarding. In the next example, every packet arriving on port 80 (the Web service port) is redirected to 10.0.0.3, which in this case would be a system running a Web server.

```
  # iptables -t nat -A PRETROUTING -i eth0 -dport 80 \
             -j DNAT --to-destination 10.0.0.3
```

With the TOS and MARK targets, you can mangle the packet to control its routing or priority. A TOS target sets the type of service for a packet, which can set the priority using criteria such as normal-service, minimize-cost, or maximize-throughput, among others.

The targets valid only for the NAT table are shown here:

SNAT	Modify source address, use **--to-source** option to specify new source address.
DNAT	Modify destination address, use **--to-destination** option to specify new destination address.
REDIRECT	Redirect a packet.
MASQUERADE	IP masquerading.
MIRROR	Reverse source and destination and send back to sender.
MARK	Modify the Mark field to control message routing.

Nat Redirection: Transparent Proxies

NAT tables can be used to implement any kind of packet redirection, a process transparent to the user. Redirection is commonly used to implement a transparent proxy. Redirection of packets is carried out with the REDIRECT target. With transparent proxies, packets received can be automatically redirected to a proxy server. For example, packets arriving on the Web service port, 80, can be redirected to the Squid Proxy service port, usually 3128. This involves a command to redirect an packet, using the REDIRECT target on the PREROUTING chain:

```
# iptables -t nat -A PREROUTING -i eth1 --dport 80 -j REDIRECT --to-port 3128
```

Packet Mangling: the Mangle Table

The *packet mangling* table is used to actually modify packet information. Rules applied specifically to this table are often designed to control the mundane behavior of packets, like routing, connection size, and priority. Rules that actually modify a packet, rather than simply redirecting or stopping it, can be used only in the mangle table. For example, the TOS target can be used directly in the mangle table to change the Type of Service field to modifying a packet's priority. A TCPMSS target could be set to control the size of a connection. The ECN target lets you work around ECN black holes, and the DSCP target will let you change DSCP bits. Several extensions such as the ROUTE extension will change a packet, in this case, rewriting its destination, rather than just redirecting it.

The mangle table is indicated with the **`-t mangle`** option. Use the following command to see what chains are listed in your mangle table:

```
iptables -t mangle  -L
```

Several mangle table targets are shown here:

TOS	Modify the Type of Service field to manage the priority of the packet.
TCPMSS	Modify the allowed size of packets for a connection, enabling larger transmissions.
ECN	Remove ECN black hole information.
DSCP	Change DSCP bits.
ROUTE	Extension TARGET to modify destination information in the packet.

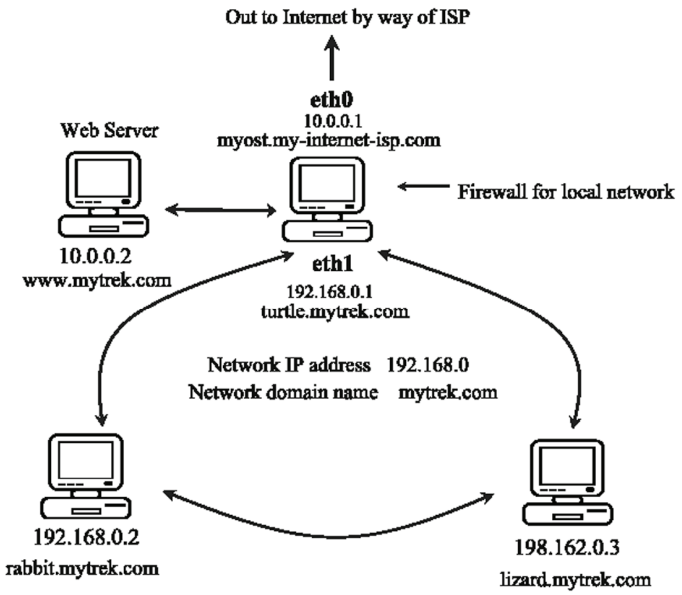


Figure 14-6: A network with a firewall

Note: The IPtables package is designed to be extensible, allowing customized targets to be added easily. This involves applying patches to the kernel and rebuilding it. See www.netfilter.org for more details, along with a listing of extended targets.

Fedora IPtables Support

Red Hat provides support for IPtables as part of their system configuration using various scripts and configuration files (see Table 14-7). When you install the RPM package for IPtables, an iptables service script is installed that will read and save IPtables commands using the **/etc/sysconfig/iptables** file. If you have set IPtables to be started up automatically when you boot your system, this file will be checked to see if it exists and is not empty. If so, IPtables will automatically read the IPtables commands that it holds. This helps to integrate IPtables more smoothly into the system setup process.

Script and Tool	Description
/etc/sysconfig/iptables	Fedora IPtables script to create IPtables rules
/etc/sysconfig/iptables.save	Fedora IPtables backup script to create IPtables rules. This is a copy made of original IPtables, when a new IPtables file is generated with the save option.
/etc/sysconfig/iptables-config	Configuration file for **/etc/rc.d/init.d/iptables** service script, containing shell variable definitions.
/etc/rc.d/init.d/iptables	Fedora iptables service script to manage IPtables rules in /etc/sysconfig/iptables.
system-config-firewall	Fedora tool for creating basic IPtables firewall rules for **/etc/syscofig/iptables**.

Table 14-7: Fedora IPtables Scripts and Tools

IPtables rules: /etc/sysconfig/iptables and system-config-firewall

The **/etc/sysconfig/iptables** script is automatically generated by **system-config-firewall**, which is run during the installation process. When you first start up your system, the **/etc/sysconfig/iptables** file will contain the IPtables rules for the configuration you selected when you ran system-config-firewall. If you run system-config-firewall again, changing your configuration, the **/etc/sysconfig/iptables** file will be overwritten with the new IPtables rules. You can access system-config-firewall as the Firewall entry on the System | Administration menu.

You can sidestep this automatic IPtables setup by simply deleting the **/etc/sysconfig/iptables** file. (Running system-config-firewall and clicking the Disable button will do the same.) Be sure you back it up first in case it has important commands. It is possible to edit the **/etc/sysconfig/iptables** file directly and enter IPtables commands, but it is not recommended. Instead, you should think of this file as holding a final installation of your IPtables commands.

The iptables service script: /etc/rc.d/init.d/iptables and /etc/sysconfig/iptables-config

You should think of the iptables service script that Fedora provides as a versatile management tool, not as a service startup script. The use of the service command for this script can

be confusing. The iptables script only manages IPtables rules, flushing, adding, or reporting them. It does not start and stop the IPtables service. If Netfilter is not running, you will need to specify that it be started up when your system boots. For this, you can use system-config-service (Services in the Server Settings window) and then select IPtables from the list of services.

The iptables service script makes use of several predefined shell parameters for specifying modules along with start and stop options. Default definitions are placed within the iptables service script, whereas corresponding custom definitions are located in a special file called **/etc/sysconfig/iptables-config**. Here the system administrator can set options like what modules to load or whether to save rules whenever IPtables is stopped, without have to edit the IPtables service script directly. For IPv6, you would use **ip6tables-config**. Table 14-8 lists the current IPtables parameters. Each time the iptables service script is used to start IPtables, it will load the modules specified in **IPTABLES_MODULES** as listed in **iptables-config**.

Parameter	Description
IPTABLES_MODULES	List of IPtables modules to load. Empty by default.
IPTABLES_MODULES_UNLOAD	Default is yes, set to yes to unload modules at service script start and stop operations.
IPTABLES_SAVE_ON_STOP	Default is no, if set to yes, will save all rules when service script stops firewall.
IPTABLES_ SAVE_ON_RESTART	Default is no, if set to yes, will save all rules when service script restarts firewall.
IPTABLES_SAVE_COUNTERS	Default is no, if set to yes, will save counters if save for stopping or restarting is enabled.
IPTABLES_STATUS_NUMERIC	Default is yes, if set to yes, will display addresses and ports numerically.

Table 14-8: Configuration Parameters for iptables-config

The service script **/etc/rc.d/init.d/iptables** supports several options with which to manage your rules. The status option displays a listing of all your current rules. The stop option will flush your current rules. Unlike stop and status, the start and save options are tied directly to the **/etc/sysconfig/iptables** file. The start option will flush your current IPtables rules and add those in the **/etc/sysconfig/iptables** file. The save option will save your current rules to the **/etc/sysconfig/iptables** file. Keep in mind that the stop and status operations work on the current IPtables rules, no matter if they were added manually on the command line, added by your own script, or added by the start option from /etc/sysconfig/iptables. The following command will list your current rules:

```
service iptables status
```

Perhaps the most effective way to think of the iptables service script is as an IPtables development tool. When creating complex firewall rules (beyond the simple set generated by **system-config-firewall**), you should first create a script and place your rules in them, as described later in the IPtables script example. Make the script executable. Any changes you need to make as you debug your firewall, you make to this script. Before you run it, run the iptables service script with the stop option to clear out any previous rules:

```
service iptables stop
```

Then run your script, as shown here for the **myfilters** script:

```
./myfilters
```

To see how the commands have been interpreted by IPtables, use the service script with the status option:

```
service iptables status
```

For any changes, edit your iptables script. Then run the service script again to clear out the old rules. Run the iptables script again, and use the status option with the service script to see how they were implemented:

```
service iptables stop
./myfilters
service iptables status
```

Saving IPtables rules

Once you are satisfied that your IPtables rules are working correctly, you can save your rules to the **/etc/sysconfig/iptables** file (for IPv6 you use **/etc/sysconfig/ip6tables**). Use the iptables service script with the save option. Now your rules will be read automatically when your system starts up. You can think of the save operation as installing your IPtables rules on your system, making them part of your system setup whenever you start your system.

```
service iptables save
```

To make changes, modify your iptables script, run the service script with stop to clear out the old rules, run the iptables script, and then use the service script with the save option to generate a new /etc/sysconfig/iptables file. A backup of the original is saved in /etc/sysconfig/iptables.save, in case to need to restore the older rules.

Instead of using the service script, you can save your rules using the iptables-save script. The recommended file to use is **/etc/iptables.rules**. The service script actually used iptables-save with the -c option to save rules to the **/etc/sysconfig/iptables** file. The -c option for iptables-save includes counters in the output (the iptables service script is designed to parse counter information along with the commands). The iptables-save command outputs rules to the standard output. To save them in a file, you must redirect the output to a file with the redirection operator, >, as shown here:

```
iptables-save -c > /etc/sysconfig/iptables
```

You can also save your rules to a file of your choosing, such as **/etc/iptables.rules**. The /etc/rc.d/init.d/iptables service script defines the IPTABLES_CONFIG variable, which holds the name of the IPtables configuration file, **/etc/sysconfig/iptables**.

```
iptables-save > /etc/iptables.rules
```

Then, to restore the rules, use the iptables-restore script to read the IPtables commands from that saved file:

```
iptables-restore < /etc/iptables.rules
```

Fedora ip6tables Support

For ip6tables, Fedora uses a different, corresponding set of supporting scripts and configuration files. ip6tables has its own service script, ip6tables, as well as its own restore and save scripts, ip6tables-save and ip6tables-restore. In their names, they have the number 6, as in /etc/sysconfig/ip6tables. The ip6tables configuration scripts and files are shown in Table 14-9.

Script and Tool	Description
/etc/sysconfig/ip6tables	Fedora ip6tables script to create IPv6 IPtables rules
/etc/sysconfig/ip6tables-config	Configuration file for /etc/rc.d/init.d/ip6tables service script, containing shell variable definitions.
/etc/rc.d/init.d/ip6tables	Fedora ip6tables service script to manage ip6tables rules in /etc/sysconfig/ip6tables.
ip6tables-save	Ip6tables save script, operates like iptables-save (see previous section).
ip6tables-restore	Ip6tables restore script, operates like iptables-restore (see previous section).

Table 14-9: Configuration File and Support Scripts for ip6tables

IPtables Scripts

Though you can enter IPtables rules from the shell command line, when you shut down your system, these commands will be lost. You will most likely need to place your IPtables rules in a script that can then be executed directly. This way you can edit and manage a complex set of rules, adding comments and maintaining their ordering.

An IPtables Script Example: IPv4

You now have enough information to create a simple IPtables script that will provide basic protection for a single system connected to the Internet. The following script, **`myfilter`**, provides an IPtables filtering process to protect a local network and a Web site from outside attacks. This example uses IPtables and IPv4 addressing. For IPv6 addressing you would use ip6tables, which has corresponding commands, except for the NAT rules, which would be implemented as mangle rules.

The script configures a simple firewall for a private network (check the IPtables HOWTO for a more complex example). If you have a local network, you could adapt this script to it. In this configuration, all remote access initiated from the outside is blocked, but two-way communication is allowed for connections that users in the network make with outside systems. In this example, the firewall system functions as a gateway for a private network whose network address is 192.168.0.0 (see Figure 14-6). The Internet address is, for the sake of this example, 10.0.0.1. The system has two Ethernet devices: one for the private network (**eth1**) and one for the Internet (**eth0**). The gateway firewall system also supports a Web server at address 10.0.0.2. Entries in this example that are too large to fit on one line are continued on a second line, with the newline quoted with a backslash. The basic rules as they apply to different parts of the network are illustrated in Figure 14-7.

myfilter

```
Firewall Gateway system IP address is 10.0.0.1 using Ethernet device eth0
# Private network address is 192.168.0.0 using Ethernet device eth1
# Web site address is 10.0.0.2
# turn off IP forwarding
echo 0 > /proc/sys/net/ipv4/ip forward
# Flush chain rules
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD
# set default (policy) rules
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

# IP spoofing, deny any packets on the internal network that have an external source address.
iptables -A INPUT -j LOG  -i eth1 \! -s 192.168.0.0/24
iptables -A INPUT -j DROP  -i eth1 \! -s 192.168.0.0/24
iptables -A FORWARD -j DROP  -i eth1 \! -s 192.168.0.0/24
# IP spoofing, deny any outside packets (any not on eth1) that have the source address of the
internal network
iptables -A INPUT -j DROP \! -i eth1 -s 192.168.0.0/24
iptables -A FORWARD -j DROP \! -i eth1 -s 192.168.0.0/24
# IP spoofing, deny any outside packets with localhost address
# (packets not on the lo interface (any on eth0 or eth1) that have the source address of
localhost)
iptables -A INPUT -j DROP  -i \! lo  -s  127.0.0.0/255.0.0.0
iptables -A FORWARD -j DROP  -i \! lo  -s  127.0.0.0/255.0.0.0

# allow all incoming messages for users on your firewall system
iptables -A INPUT -j ACCEPT  -i lo

# allow  communication to the Web server (address 10.0.0.2), port www
iptables -A INPUT  -j ACCEPT -p tcp -i eth0  --dport www -s 10.0.0.2
# Allow  established connections from Web servers to internal network
iptables -A INPUT -m state --state ESTABLISHED,RELATED -i eth0 -p tcp  --sport www -s
10.0.0.2 -d 192.168.0.0/24  -j ACCEPT
# Prevent new  connections from Web servers to internal network
iptables -A OUTPUT -m state --state  NEW -o eth0 -p tcp  --sport www -d 192.168.0.0/24  -j
DROP

# allow established and related outside communication to your system
# allow outside communication to the firewall, except for ICMP packets
iptables -A INPUT -m state --state ESTABLISHED,RELATED -i eth0 -p \! icmp -j ACCEPT
# prevent outside initiated connections
iptables -A INPUT -m state --state NEW -i eth0 -j DROP
iptables -A FORWARD -m state --state NEW -i eth0 -j DROP
# allow all local communication to and from the firewall on eth1  from the local network
iptables -A INPUT -j ACCEPT -p all -i eth1 -s 192.168.0.0/24
# Set up masquerading to allow internal machines access to outside network
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
# Accept ICMP Ping and Destination unreachable messages
# Others will be rejected by INPUT and OUTPUT DROP policy
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 --icmp-type  echo-reply -d 10.0.0.1
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 --icmp-type  echo-request -d 10.0.0.1
iptables -A INPUT -j ACCEPT -p icmp -i eth0 --icmp-type  destination-unreachable -d 10.0.0.1
# Turn on IP Forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
```

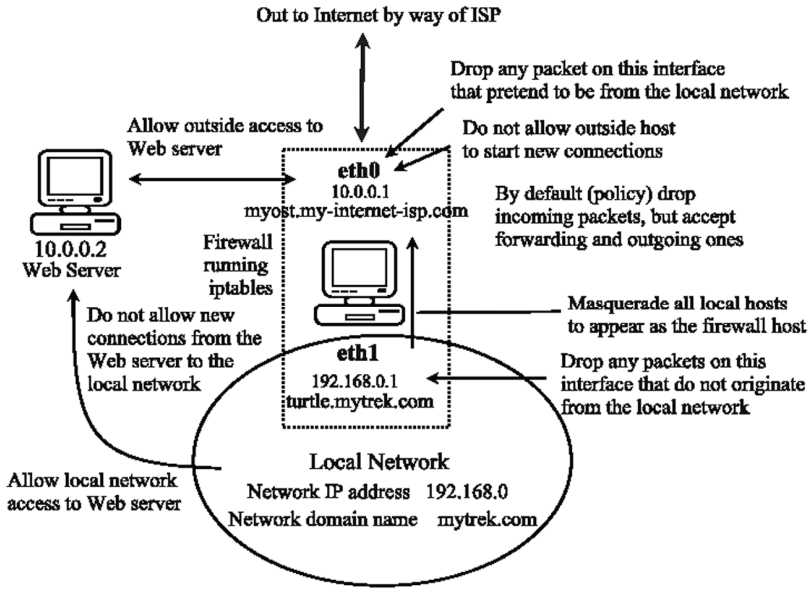


Figure 14-7: Firewall rules applied to a local network example

Initially, in the script you would clear your current IPtables with the flush option (**-F**), and then set the policies (default targets) for the non-user-defined rules. IP forwarding should also be turned off while the chain rules are being set:

```
echo 0 > /proc/sys/net/ipv4/ip_forward
```

Drop Policy

First, a DROP policy is set up for INPUT and FORWARD built-in IP chains. This means that if a packet does not meet a criterion in any of the rules to let it pass, it will be dropped. Then both IP spoofing attacks and any attempts from the outside to initiate connections (SYN packets) are rejected. Outside connection attempts are also logged. This is a very basic configuration that can easily be refined to your own needs by adding IPtables rules.

```
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
```

IP Spoofing

One way to protect the private network from the IP spoofing of any packets is to check for any outside addresses on the Ethernet device dedicated to the private network. In this example, any packet on device **eth1** (dedicated to the private network) whose source address is not that of the private network (**! -s 192.168.0.0**) is denied. Also, check to see if any packets coming from the outside are designating the private network as their source. In this example, any packets with the source address of the private network on any Ethernet device other than for the private network (**eth1**) are denied. The same strategy can be applied to the local host.

```
# IP spoofing, deny any packets on the internal network
# that has an external source address.
iptables -A INPUT -j LOG  -i eth1 \! -s 192.168.0.0/24
iptables -A INPUT -j DROP  -i eth1 \! -s 192.168.0.0/24
iptables -A FORWARD -j DROP  -i eth1 \! -s 192.168.0.0/24
# IP spoofing, deny any outside packets (any not on eth1)
# that have the source address of the internal network
iptables -A INPUT -j DROP \! -i eth1 -s 192.168.0.0/24
iptables -A FORWARD -j DROP \! -i eth1 -s 192.168.0.0/24
# IP spoofing, deny any outside packets with localhost address
# (packets not on the lo interface (any on eth0 or eth1)
# that have the source address of localhost)
iptables -A INPUT -j DROP  -i \! lo  -s  127.0.0.0/255.0.0.0
iptables -A FORWARD -j DROP  -i \! lo  -s  127.0.0.0/255.0.0.0
```

Then, you would set up rules to allow all packets sent and received within your system (localhost) to pass.

```
iptables -A INPUT -j ACCEPT  -i lo
```

Server Access

For the Web server, you want to allow access by outside users but block access by anyone attempting to initiate a connection from the Web server into the private network. In the next example, all messages are accepted to the Web server, but the Web server cannot initiate contact with the private network. This prevents anyone from breaking into the local network through the Web server, which is open to outside access. Established connections are allowed, permitting the private network to use the Web server.

```
# allow  communication to the Web server (address 10.0.0.2), port www
iptables -A INPUT  -j ACCEPT -p tcp -i eth0  --dport www -s 10.0.0.2
# Allow  established connections from Web servers to internal network
iptables -A INPUT -m state --state ESTABLISHED,RELATED -i eth0 \
   -p tcp  --sport www -s 10.0.0.2 -d 192.168.0.0/24  -j ACCEPT
# Prevent new  connections from Web servers to internal network
iptables -A OUTPUT -m state --state  NEW -o eth0 -p tcp \
  --sport www -d 192.168.0.1.0/24  -j DROP
```

Firewall Outside Access

To allow access by the firewall to outside networks, you allow input by all packets except for ICMP packets. These are handled later. The firewall is specified by the firewall device, **eth0**. First your firewall should allow established and related connections to proceed, as shown here. Then you would block outside access as described later.

```
# allow outside communication to the firewall,
# except for ICMP packets
iptables -A INPUT -m state --state ESTABLISHED,RELATED \
         -i eth0 -p \! icmp -j ACCEPT
```

Blocking Outside Initiated Access

To prevent outsiders from initiating any access to your system, create a rule to block access by SYN packets from the outside using the **`state`** option with NEW. Drop any new

connections on the **eth0** connection (assumes only **eth0** is connected to the Internet or outside network).

```
# prevent outside initiated connections
iptables -A INPUT -m state --state NEW -i eth0 -j DROP
iptables -A FORWARD -m state --state NEW -i eth0 -j DROP
```

Local Network Access

To allow interaction by the internal network with the firewall, you allow input by all packets on the internal Ethernet connection, **eth1**. The valid internal network addresses are designated as the input source.

```
iptables -A INPUT -j ACCEPT -p all -i eth1 -s 192.168.0.0/24
```

Listing Rules

A listing of these **iptables** options shows the different rules for each option, as shown here:

```
# iptables -L
Chain INPUT (policy DROP)
target   prot opt source          destination
LOG      all  -- !192.168.0.0/24  anywhere        LOG level warning
DROP     all  -- !192.168.0.0/24  anywhere
DROP     all  --  192.168.0.0/24  anywhere
DROP     all  --  127.0.0.0/8     anywhere
ACCEPT   all  --  anywhere        anywhere
ACCEPT   tcp  --  10.0.0.2        anywhere        tcp dpt:http
ACCEPT   tcp  --  10.0.0.2        192.168.0.0/24  state RELATED,ESTABLISHED
                                                          tcp spt:http
ACCEPT  !icmp --  anywhere        anywhere        state RELATED,ESTABLISHED
DROP     all  --  anywhere        anywhere        state NEW
ACCEPT   all  --  192.168.0.0/24  anywhere
ACCEPT   icmp --  anywhere        10.0.0.1        icmp echo-reply
ACCEPT   icmp --  anywhere        10.0.0.1        icmp echo-request
ACCEPT   icmp --  anywhere        10.0.0.1        icmp destination-unreachable
Chain FORWARD (policy ACCEPT)
target   prot opt source          destination
DROP     all  -- !192.168.0.0/24  anywhere
DROP     all  --  192.168.0.0/24  anywhere
DROP     all  --  127.0.0.0/8     anywhere
DROP     all  --  anywhere        anywhere        state NEW

Chain OUTPUT (policy ACCEPT)
target   prot opt source       destination
DROP       tcp  --  anywhere     192.168.0.0/24  state NEW tcp spt:http

# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target      prot opt source       destination
Chain POSTROUTING (policy ACCEPT)
target      prot opt source       destination
MASQUERADE  all  --  anywhere      anywhere
```

```
Chain OUTPUT (policy ACCEPT)
target     prot opt source       destination
```

User-Defined Rules

For more complex rules, you may want to create your own chain to reduce repetition. A common method is to define a user chain for both INPUT and FORWARD chains, so that you do not have to repeat DROP operations for each. Instead, you would have only one user chain that both FORWARD and INPUT chains would feed into for DROP operations. Keep in mind that both FORWARD and INPUT operations may have separate rules in addition to the ones they share. In the next example, a user-defined chain called arriving is created. The chain is defined with the **-N** option at the top of the script:

```
iptables -N arriving
```

A user chain has to be defined before it can be used as a target in other rules. So you have to first define and add all the rules for that chain, and then use it as a target. The arriving chain is first defined and its rules added. Then, at the end of the file, it is used as a target for both the INPUT and FORWARD chains. The INPUT chain lists rules for accepting packets, whereas the FORWARD chain has an ACCEPT policy that will accept them by default.

```
iptables -N arriving
iptables -F arriving
# IP spoofing, deny any packets on the internal network
# that has an external source address.
iptables -A arriving -j LOG  -i eth1 \! -s 192.168.0.0/24
iptables -A arriving -j DROP  -i eth1 \! -s 192.168.0.0/24
iptables -A arriving -j DROP \! -i eth1 -s 192.168.0.0/24
……
# entries at end of script
iptables -A INPUT -j arriving
iptables -A FORWARD -j arriving
```

A listing of the corresponding rules is shown here:

```
Chain INPUT (policy DROP)
target    prot opt source          destination
arriving  all  --  0.0.0.0/0         0.0.0.0/0
Chain FORWARD (policy ACCEPT)
target    prot opt source          destination
arriving  all  --  0.0.0.0/0         0.0.0.0/0
Chain arriving (2 references)
target    prot opt source          destination
LOG       all  -- !192.168.0.0/24   0.0.0.0/0       LOG flags 0 level 4
DROP      all  -- !192.168.0.0/24   0.0.0.0/0
DROP      all  --  192.168.0.0/24   0.0.0.0/0
```

For rules where chains may differ, you will still need to enter separate rules. In the **myfilter** script, the FORWARD chain has an ACCEPT policy, allowing all forwarded packets to the local network to pass through the firewall. If the FORWARD chain had a DROP policy, like the INPUT chain, then you may need to define separate rules under which the FORWARD chain could accept packets. In this example, the FORWARD and INPUT chains have different rules for accepting packets on the **eth1** device. The INPUT rule is more restrictive. To enable the local

network to receive forwarded packets through the firewall, you could enable forwarding on its device using a separate FORWARD rule, as shown here:

```
iptables -A FORWARD -j ACCEPT -p all -i eth1
```

The INPUT chain would accept packets only from the local network.

```
iptables -A INPUT -j ACCEPT -p all -i eth1 -s 192.168.0.0/24
```

Masquerading Local Networks

To implement masquerading, where systems on the private network can use the gateway's Internet address to connect to Internet hosts, you create a NAT table (**`-t nat`**) POSTROUTING rule with a MASQUERADE target.

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Controlling ICMP Packets

In addition, to allow ping and destination-reachable ICMP packets, you enter INPUT rules with the firewall as the destination. To enable ping operations, you use both echo-reply and echo-request ICMP types, and for destination unreachable, you use the destination-unreachable type.

```
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 --icmp-type \
   echo-reply -d 10.0.0.1
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 --icmp-type \
   echo-request -d 10.0.0.1
iptables -A INPUT -j ACCEPT -p icmp -i eth0 --icmp-type \
   destination-unreachable -d 10.0.0.1
```

At the end, IP forwarding is turned on again.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Simple LAN Configuration

To create a script to support a simple LAN without any Internet services like Web servers, you would just not include rules for supporting those services. You would still need FORWARD and POSTROUTING rules for connecting your local hosts to the Internet, as well as rules governing interaction between the hosts and the firewall. To modify the example script to support a simple LAN without the Web server, just remove the three rules governing the Web server. Leave everything else the same.

LAN Configuration with Internet Services on the Firewall System

Often, in the same system that functions as a firewall is also used to run Internet servers, like Web and FTP servers. In this case the firewall rules are applied to the ports used for those services. The example script dealt with a Web server running on a separate host system. If the Web server were instead running on the firewall system, you would apply the Web server firewall rules to the port that the Web server uses. Normally the port used for a Web server is 80. In the following example, the IPtables rules for the Web server have been applied to port www, port 80, on the

firewall system. The modification simply requires removing the old Web server host address references, 10.0.0.2.

```
# allow  communication to the Web server, port www (port 80)
iptables -A INPUT  -j ACCEPT -p tcp -i eth0  --dport www
# Allow  established connections from Web servers to internal network
iptables -A INPUT -m state --state ESTABLISHED,RELATED -i eth0 \
   -p tcp  --sport www -d 192.168.0.0/24  -j ACCEPT
# Prevent new  connections from Web servers to internal network
iptables -A OUTPUT -m state --state  NEW -o eth0 -p tcp \
  --sport www -d 192.168.0.1.0/24 -j DROP
```

Similar entries could be set up for an FTP server. Should you run several Internet services, you could use a user-defined rule to run the same rules on each service, rather than repeating three separate rules per service. Working from the example script, you would use two defined rules, one for INPUT and one for OUTPUT, controlling incoming and outgoing packets for the services.

```
iptables -N inputservice
iptables -N outputservice
iptables -F inputservice
iptables -F outputservice
# allow  communication to the service
iptables -A inputservice  -j ACCEPT -p tcp -i eth0
# Allow  established connections from the service to internal network
iptables -A inputservice -m state --state ESTABLISHED,RELATED -i eth0 \
   -p tcp  -d 192.168.0.0/24  -j ACCEPT
# Prevent new  connections from service to internal network
iptables -A outputservice -m state --state  NEW -o eth0 -p tcp \
  -d 192.168.0.1.0/24 -j DROP
.....
# Run rules for the Web server, port www (port 80)
iptables -A INPUT  --dport www -j inputservice
iptables -A INPUT  --dport www -j outputservice
# Run rules for the FTP server, port ftp (port 21)
iptables -A OUTPUT  --dport ftp -j inputservice
iptables -A OUTPUT  --dport ftp -j outputservice
```

IP Masquerading

On Linux systems, you can set up a network in which you can have one connection to the Internet, which several systems on your network can use. This way, using only one IP address, several different systems can connect to the Internet. This method is called *IP masquerading,* where a system masquerades as another system, using that system's IP address. In such a network, one system is connected to the Internet with its own IP address, while the other systems are connected on a local area network (LAN) to this system. When a local system wants to access the network, it masquerades as the Internet-connected system, borrowing its IP address.

IP masquerading is implemented on Linux using the IPtables firewall tool. In effect, you set up a firewall, which you then configure to do IP masquerading. You can find out more information on IP masquerading from the Linux Masquerade HOWTO files at the Linux Documentation Project, **http://tldp.org**, where you'll find both an IP-Masquerade-HOWTO and a Masquerading-Simple-HOWTO. They provide detailed, step-by-step guides to setting up IP

masquerading on your system. IP masquerading must be supported by the kernel before you can use it.

With IP masquerading, as implemented on Linux systems, the machine with the Internet address is also the firewall and gateway for the LAN of machines that use the firewall's Internet address to connect to the Internet. Firewalls that also implement IP masquerading are sometimes referred to as *MASQ gates.* With IP masquerading, the Internet-connected system (the firewall) listens for Internet requests from hosts on its LAN. When it receives one, it replaces the requesting local host's IP address with the Internet IP address of the firewall and then passes the request out to the Internet, as if the request were its own. Replies from the Internet are then sent to the firewall system. The replies the firewall receives are addressed to the firewall using its Internet address. The firewall then determines the local system to whose request the reply is responding. It then strips off its IP address and sends the response on to the local host across the LAN. The connection is transparent from the perspective of the local machines. They appear to be connected directly to the Internet.

Masquerading Local Networks

IP masquerading is often used to allow machines on a private network to access the Internet. These could be machines in a home network or a small LAN, such as for a small business. Such a network might have only one machine with Internet access, and as such, only the one Internet address. The local private network would have IP addresses chosen from the private network allocations (10., 172.16., or 192.168.). Ideally, the firewall has two Ethernet cards: one for an interface to the LAN (for example, **eth1**) and one for an interface to the Internet, such as **eth0** (for dial-up ISPs, this would be **ppp0** for the modem). The card for the Internet connection (**eth0**) would be assigned the Internet IP address. The Ethernet interface for the local network (**eth1**, in this example) is the firewall Ethernet interface. Your private LAN would have a network address like 192.168.0. Its Ethernet firewall interface (**eth1**) would be assigned the IP address 192.168.0.1. In effect, the firewall interface lets the firewall operate as the local network's gateway. The firewall is then configured to masquerade any packets coming from the private network. Your LAN needs to have its own domain name server, identifying the machines on your network, including your firewall. Each local machine needs to have the firewall specified as its gateway. Try not to use IP aliasing to assign both the firewall and Internet IP addresses to the same physical interface. Use separate interfaces for them, such as two Ethernet cards, or an Ethernet card and a modem (**ppp0**).

Masquerading NAT Rules

In Netfilter, IP masquerading is a NAT operation and is not integrated with packet filtering as in IP Chains. IP masquerading commands are placed on the NAT table and treated separately from the packet-filtering commands. Use IPtables to place a masquerade rule on the NAT table. First reference the NAT table with the **`-t nat`** option. Then add a rule to the POSTROUTING chain with the **`-o`** option specifying the output device and the **`-j`** option with the MASQUERADE command.

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

IP Forwarding

The next step is to turn on IP forwarding, either manually or by setting the **`net.ipv4.ip_forward`** variable in the **/etc/sysctl.conf** file and running **`sysctl`** with the **`-p`** option.

IP forwarding will be turned off by default. For IPv6, use **net.ipv6.conf.all.forwarding**. The **/etc/sysctl.conf** entries are shown here:

```
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
```

You then run **sysctl** with the **-p** option.

```
sysctl -p
```

You can directly change the respective forwarding files with an **echo** command as shown here:

```
echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
```

For IPv6, you would to use the forwarding file in the corresponding **/proc/sys/net/ipv6** directory, **conf/all/forwarding**.

```
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

Masquerading Selected Hosts

Instead of masquerading all local hosts as the single IP address of the firewall/gateway host, you could use the NAT table to rewrite addresses for a few selected hosts. Such an approach is often applied to setups where you want several local hosts to appear as Internet servers. Using the DNAT and SNAT targets, you can direct packets to specific local hosts. You would use rules on the PREROUTING and POSTROUTING chains to direct input and output packets.

For example, the Web server described in the previous example could have been configured as a local host to which a DNAT target could redirect any packets originally received for 10.0.0.2. Say the Web server was set up on 192.168.0.5. It could appear as having the address 10.0.0.2 on the Internet. Packets sent to 10.0.0.2 would be rewritten and directed to 192.168.0.5 by the NAT table. You would use the PREROUTING chain with the **-d** option to handle incoming packets and POSTROUTING with the **-s** option for outgoing packets.

```
iptables -t nat -A PREROUTING -d 10.0.0.2 --to-destination 192.168.0.5 -j DNAT
iptables -t nat -A POSTROUTING -s 192.168.0.5 --to-source 10.0.0.2 -j SNAT
```

Tip: Bear in mind that with IPtables, masquerading is not combined with the FORWARD chain, as it is with IP Chains. So, if you specify a DROP policy for the FORWARD chain, you will also have to specifically enable FORWARD operation for the network that is being masqueraded. You will need both a POSTROUTING rule and a FORWARD rule.

Part 3: File Systems and Devices

fedora

15. File System management

File Systems

Filesystem Hierarchy Standard

Journaling

Mounting File Systems Automatically: /etc/fstab

Mounting File Systems Manually: mount and umount

Creating File Systems: mkfs, mke2fs, mkswap, parted, and fdisk

CD-ROM and DVD ROM Recording

Mono and .NET Support

Files reside on physical storage devices such as hard drives, CD-ROMs, or floppy disks. The files on each storage device are organized into a file system. The storage devices on your Linux system are presented as a collection of file systems that you can manage. When you want to add a new storage device, you need to format it as a file system and then attach it to your Linux file structure. Hard drives can be divided into separate storage devices called *partitions,* each of which has its own file system. You can perform administrative tasks on your file systems, such as backing them up, attaching or detaching them from your file structure, formatting new devices or erasing old ones, and checking a file system for problems.

To access files on a device, you attach its file system to a specified directory. This is called *mounting* the file system. For example, to access files on a USB drive, you first mount its file system to a particular directory. With Linux, you can mount a number of different types of file systems. You can even access a Windows hard drive partition or tape drive, as well as file systems on a remote server.

Linux file systems support *journaling,* which allows your system to recover from a crash or interruption easily. The ext3, ReiserFS, XFS, and JFS (IBM) file systems maintain a record of file and directory changes, called a *journal,* which can be used to recover files and directories in use when a system suddenly fails due to unforeseen events such as power interruptions. Most distributions currently use the **ext3** file system as their default, though you also have the option of using ReiserFS or JFS, an independently developed journaling system. Also available is the new **ext4** file system which provides better access to very large files like video.

Your Linux system is capable of handling any number of storage devices that may be connected to it. You can configure your system to access multiple hard drives, partitions on a hard drive, CD-ROM discs, DVDs, floppy disks, and even tapes. You can elect to attach these storage components manually or have them automatically mount when you boot. Automatic mounts are handled by configuring the **/etc/fstab** file. For example, the main partitions holding your Linux system programs are automatically mounted whenever you boot, whereas a floppy disk can be manually mounted when you put one in your floppy drive, though even these can also be automatically mounted. Removable storage devices like CD-ROMs, as well as removable devices like USB cameras and printers, are now handled by udev and the Hardware Abstract Layer (HAL), as described in Chapter 17 and partially discussed here. HAL and udev also will automatically mount any Windows **ntfs** file systems on your local hard drives. Remote files systems are mounted using Samba and NFS servers. All your mounted file systems will be browsable from your GNOME and KDE desktops.

File Systems

Although all the files in your Linux system are connected into one overall directory tree, parts of that tree may reside on different storage devices such as hard drives or CD-ROMs. Files on a particular storage device are organized into what is referred to as a *file system.* A file system is a formatted device, with its own tree of directories and files. Your Linux directory tree may encompass several file systems, each on different storage devices. On a hard drive with several partitions, you would have a file system for each partition. The files themselves are organized into one seamless tree of directories, beginning from the root directory. For example, if you attach a CD-ROM to your system, a pathname will lead directly from the root directory on your main root hard disk partition's file system to the files in the CD-ROM file system.

A file system has its files organized into its own directory tree. You can think of this as a *subtree* that must be attached to the main directory tree. The tree remains separate from your system's directory tree until you specifically connect it. For example, a floppy disk with Linux files has its own tree of directories. You need to attach this subtree to the main tree on your hard drive partition. Until they are attached, you cannot access the files on your floppy disk.

Directory	Function
/	Begins the file system structure—called the root.
/boot	Holds the kernel image files and associated boot information and files.
/home	Contains users' home directories.
/sbin	Holds administration-level commands and any commands used by the root user.
/dev	Holds dynamically generated file interfaces for devices such as the terminal and the printer (see "udev: Device Files" in Chapter 17).
/etc	Holds system configuration files and any other system files.
/etc/opt	Holds system configuration files for applications in **/opt**.
/etc/X11	Holds system configuration files for the X Window System and its applications.
/bin	Holds the essential user commands and utility programs.
/lib	Holds essential shared libraries and kernel modules.
/lib/modules	Holds the kernel modules.
/media	Used to hold directories for mounting media-based removable file systems, like CD-ROMs, floppy disks, USB card readers, and digital cameras.
/mnt	Used to hold directories for additional file systems such as hard disks.
/opt	Holds added software applications (for example, KDE on some distributions).
/proc	Process directory, a memory-resident directory containing files used to provide information about the system.
/sys	The sysfs file system for kernel objects, listing supported kernel devices and modules.
/tmp	Holds temporary files.
/usr	Holds those files and commands used by the system; this directory breaks down into several subdirectories.
/var	Holds files that vary, such as mailbox, Web, and FTP files.

Table 15-1: Linux File System Directories

Filesystem Hierarchy Standard

Linux organizes its files and directories into one overall interconnected tree, beginning from the root directory and extending down to system and user directories. The organization and layout for the system directories are determined by the Filesystem Hierarchy Standard (FHS). The FHS provides a standardized layout that all Linux distributions should follow in setting up their system directories.

For example, there must be an **/etc** directory to hold configuration files and a **/dev** directory for device files. You can find out more about FHS, including the official documentation, at **http://www.pathname.com/fhs**. Linux distributions, developers, and administrators all follow the FHS to provide a consistent organization to the Linux file system.

Linux uses a number of specifically named directories for specialized administrative tasks. All these directories are at the very top level of your main Linux file system, the file system root directory, represented by a single slash, /. For example, the **/dev** directory holds device files, and the **/home** directory holds the user home directories and all their user files. You have access to these directories and files only as the system administrator (though users normally have read-only access). You need to log in as the root user (**su** command, or as the **root** user on runlevel **3**), placing yourself in a special root user administrative directory called **/root**. From here, you can access any directory on the Linux file system, both administrative and user.

Root Directory: /

The subdirectories held in the root directory, /, are listed in Table 15-1, along with other useful subdirectories. Directories that you may commonly access as an administrator are the **/etc** directory, which holds configuration files; the **/dev** directory, which holds dynamically generated device files; and the **/var** directory, which holds server data files for DNS, Web, mail, and FTP servers, along with system logs and scheduled tasks. For managing different versions of the kernel, you may need to access the **/boot** and **/lib/modules** directories as well as **/usr/src/linux**. The **/boot** directory holds the kernel image files for any new kernels you install, and the **/lib/modules** directory holds modules for your different kernels.

Tip: With Linux you can mount file systems of different types, including those created by other operating systems, including Windows, Unix, and OSX. Within Linux a variety of file systems are supported, including several journaling systems like ReiserFS and ext3.

System Directories

Your Linux directory tree contains certain directories whose files are used for different system functions. For basic system administration, you should be familiar with the system program directories where applications are kept, the system configuration directory (**/etc**) where most configuration files are placed, and the system log directory (**/var/log**) that holds the system logs, recording activity on your system. Table 15-2 lists the system directories.

Program Directories

Directories with **bin** in the name are used to hold programs. The **/bin** directory holds basic user programs, such as login, shells (BASH, TCSH, and zsh), and file commands (`cp`, `mv`, `rm`, `ln`, and so on). The **/sbin** directory holds specialized system programs for such tasks as file system

management (**fsck**, **fdisk**, **mkfs**) and system operations like shutdown and startup (**init**). The **/usr/bin** directory holds program files designed for user tasks. The **/usr/sbin** directory holds user-related system operation, such as **useradd** for adding new users. The **/lib** directory holds all the libraries your system makes use of, including the main Linux library, **libc**, and subdirectories such as **modules**, which holds all the current kernel modules. It will also hold some system controlled configuration files, like **/lib/udev**.

Directory	Description
/bin	System-related programs
/sbin	System programs for specialized tasks
/lib	System libraries
/etc	Configuration files for system and network services and applications
/home	The location of user home directories and server data directories, such as Web and FTP site files
/media	The location where removable media file systems like CD-ROMs and floppy disks are mounted
/var	The location of system directories whose files continually change, such as logs, printer spool files, and lock files
/usr	User-related programs and files. Includes several key subdirectories, such as **/usr/bin**, **/usr/X11**, and **/usr/share/doc**
/usr/bin	Programs for users
/dev	Device files
/sys	The sysfs file system with device information for kernel-supported devices on your system
/usr/X11	X Window System configuration files
/usr/share	Shared files
/usr/share/doc	Documentation for applications
/usr/share/hal	Configuration for HAL removable devices
/etc/udev **/lib/udev**	Configuration for device files, udev
/tmp	Directory for system temporary files

Table 15-2: System Directories

Configuration Directories and Files

When you configure different elements of your system, such as user accounts, applications, servers, or network connections, you make use of configuration files kept in certain system directories. Configuration files are placed in the **/etc** directory.

The /usr Directory

The **/usr** directory contains a multitude of important subdirectories used to support users, providing applications, libraries, and documentation. The **/usr/bin** directory holds numerous user-accessible applications and utilities; **/usr/sbin** hold user-accessible administrative utilities. The **/usr/share** directory holds architecture-independent data that includes an extensive number of subdirectories, including those for documentation, such as **man**, **info**, and **doc** files. Table 15-3 lists the subdirectories of the **/usr** directory.

Directory	Description
/usr/bin	Holds most user commands and utility programs.
/usr/sbin	Holds administrative applications.
/usr/lib	Holds libraries for applications, programming languages, desktops, and so on.
/usr/games	Holds games and educational programs.
/usr/include	Holds C programming language header files (**.h**).
/usr/local	Holds locally installed software.
/usr/share	Holds architecture-independent data such as documentation.
/usr/src	Holds source code, including the kernel source code.

Table 15-3: /usr Directories

The /media Directory

The **/media** directory is used for mountpoints for removable media like CD-ROM, DVD, floppy, or Zip drives, as well as for other media-based file systems such as USB card readers, cameras, and MP3 players. These are file systems you may be changing frequently, unlike partitions on fixed disks. Most Linux systems use the Hardware Abstraction Layer (HAL) to dynamically manage the creation, mounting, and device assignment of these devices. As instructed by HAL, this tool will create floppy, CD-ROM, storage card, camera, and MP3 player subdirectories in **/media** as needed. The default subdirectory for mounting is **/media/disk**. Additional drives have an number attached to their name.

The /mnt Directory

The **/mnt** directory is usually used for mountpoints for other mounted file systems such as Windows partitions. You can create directories for any partitions you want to mount, such as **/mnt/windows** for a Windows partition.

The /home Directory

The **/home** directory holds user home directories. When a user account is set up, a home directory is set up here for that account, usually with the same name as the user. As the system administrator, you can access any user's home directory, and so you have control over that user's files.

The /var Directory

The **/var** directories are designed to hold data that changes with the normal operation of the Linux system. For example, spool files for documents that you are printing are kept here. A spool file is created as a temporary printing file and is removed after printing. Other files, such as system log files, are changed constantly. Table 15-4 lists the subdirectories of the **/var** directory.

The **/var** directory holds subdirectories for tasks whose files change frequently, such as lock files, log files, Web server files, or printer spool files. For example, the **/var** directory holds server data directories, such as **/var/www** for the Apache Web server Web site files or **/var/ftp** for your FTP site files, as well as **/var/named** for the DNS server. The **/var/tmp** directory is simply a directory to hold any temporary files programs may need to perform a particular task.

Directory	Description
/var/account	Processes accounting logs.
/var/cache	Holds application cache data for Man pages, Web proxy data, fonts, or application-specific data.
/var/games	Holds varying games data.
/var/lib	Holds state information for particular applications.
/var/local	Used for data that changes for programs installed in **/usr/local**.
/var/lock	Holds lock files that indicate when a particular program or file is in use.
/var/log	Holds log files such as **/var/log/messages** that contain all kernel and system program messages.
/var/mail	Holds user mailbox files, link to **/var/spool/mail**.
/var/opt	Holds variable data for applications installed in **/opt**.
/var/run	Holds information about the system's running processes.
/var/spool	Holds applications' spool data such as that for mail, news, and printer queues, as well as **`cron`** and **`at`** jobs.
/var/tmp	Holds temporary files that should be preserved between system reboots.
/var/yp	Holds Network Information Service (NIS) data files.
/var/www	Holds Web server Web site files.
/var/ftp	Holds FTP server FTP files.
/var/named	Holds DNS server domain configuration files.

Table 15-4: /var Subdirectories

The /proc File System

The **/proc** file system is a special file system that is generated in system memory. It does not exist on any disk. **/proc** contains files that provide important information about the state of your

system. For example, **/proc/cpuinfo** holds information about your computer's CPU processor. **/proc/devices** lists those devices currently configured to run with your kernel. **/proc/filesystems** lists the file systems. **/proc** files are really interfaces to the kernel, obtaining information from the kernel about your system. Table 15-5 lists the **/proc** subdirectories and files.

Like any file system, **/proc** has to be mounted. The **/etc/fstab** file will have a special entry for **/proc** with a file system type of proc and no device specified.

Tip: You can use **sysctl**, the Kernel Tuning tool, to set proc file values you are allowed to change, like the maximum number of files, or turning on IP forwarding.

File	Description
/proc/*num*	There is a directory for each process labeled by its number. **/proc/1** is the directory for process 1.
/proc/cpuinfo	Contains information about the CPU, such as its type, make, model, and performance.
/proc/devices	Lists the device drivers configured for the currently running kernel.
/proc/dma	Displays the DMA channels currently used.
/proc/filesystems	Lists file systems configured into the kernel.
/proc/interrupts	Displays the interrupts in use.
/proc/ioports	Shows the I/O ports in use.
/proc/kcore	Holds an image of the physical memory of the system.
/proc/kmsg	Contains messages generated by the kernel.
/proc/loadavg	Lists the system load average.
/proc/meminfo	Displays memory usage.
/proc/modules	Lists the kernel modules currently loaded.
/proc/net	Lists status information about network protocols.
/proc/stat	Contains system operating statistics, such as page fault occurrences.
/proc/uptime	Displays the time the system has been up.
/proc/version	Displays the kernel version.

Table 15-5: /proc Subdirectories and Files

The sysfs File System: /sys

The **sysfs** file system is a virtual file system that provides the a hierarchical map of your kernel-supported devices such as PCI devices, buses, and block devices, as well as supporting kernel modules. The **classes** subdirectory will list all your supported devices by category, such as net and sound devices. With **sysfs** your system can easily determine the device file a particular device is associated with. This is very helpful for managing removable devices as well as dynamically configuring and managing devices as HAL and udev do. The **sysfs** file system is used by udev to dynamically generate needed device files in the **/dev** directory, as well as by HAL to

manage removable device files and support as needed (HAL technically provides information only about devices, though it can use tools to dynamically change configurations as needed). The **/sys** file system type is **sysfs**. The **/sys** subdirectories organize your devices into different categories. This file system can be used by **systool** to display a listing of your installed devices. **systool** is part of the **sysfsutils** package. The following example will list all your system devices.

```
systool
```

Like **/proc**, the **/sys** directory resides only in memory, but you still need to mount in the **/etc/fstab** file.

```
none    /sys       sysfs     defaults   0       0
```

Device Name	Description
sd	Hard drives (IDE, SATA, and SCSI, originally used for just SCSI)
scd	SATA and SCSI DVD/CD-ROM drives
fd	Floppy disks
st	SCSI tape drives
nst	SCSI tape drives, no rewind
ht	IDE tape drives
tty	Terminals
lp	Printer ports
pty	Pseudoterminals (used for remote logins)
midi	Midi ports
ttyS	Serial ports
md	RAID devices
cdrom	Link to your CD-ROM device file
cdrw	Link to your CD-R or CD-RW device
modem	Link to your modem device file
floppy	Link to your floppy device file
tape	Link to your tape device file
scanner	Link to your scanner device file

Table 15-6: Device Name Prefixes

Device Files: /dev, udev, and HAL

To mount a file system, you have to specify its device name. The interfaces to devices that may be attached to your system are provided by special files known as *device files.* The names of these device files are the device names. Device files are located in the **/dev** directories and usually have abbreviated names ending with the number of the device. For example, **fd0** may reference the

first floppy drive attached to your system. The prefix **sd** references hard drives, so **sda2** would reference the second partition on first hard drive. The **sd** prefix is used for all hard drives, including IDE ATA, Serial ATA, and SCSI hard drives. In most cases, you can use the **man** command with a prefix to obtain more detailed information about this kind of device. For example, **man fd** displays the Man page for Floppy devices (you still use **man hd** for information on IDE ATA drives). A complete listing of all device names can be found in the **device-list** file at the **www.kernel.org** Web site (**www.kernel.org/pub/linux/docs/device-list/devices.txt**). Table 15-6 lists several of the commonly used device names.

Tip: The device Man pages for hard disks still use their original prefix names, even though **sd** is now used for all hard disks. The Man page **sd** displays information about SCSI hard disks and **hd** shows information about IDE hard drives.

udev and HAL

Device files are no longer handled in a static way. Instead they are now dynamically generated as needed. Previously a device file was created for each possible device, leading to a very large number of device files in the **/etc/dev** directory. Now, your system will detect only those devices it uses and create device files for those only, giving you a much smaller listing of device files. The tool used to detect and generate device files is udev, user devices. Each time your system is booted, udev will automatically detect your devices and generate device files for them in the **/etc/dev** directory. This means that the **/etc/dev** directory and its files are recreated each time you boot. It is a dynamic directory, no longer static. To manage these device files, you need to use udev configuration files located in the **/etc/udev** and **/lib/udev** directories. This means that udev is able to also dynamically manage all removable devices; udev will generate and configure devices files for removable devices as they are attached, and then remove these files when the devices are removed. In this sense, all devices are now considered hotplugged, with fixed devices simply being hotplugged devices that are never removed.

As **/etc/dev** is now dynamic, any changes you would make manually to the **/etc/dev** directory will be lost when you reboot. This includes the creation of any symbolic links like **/dev/cdrom** that many software applications use. Instead, such symbolic links have to be configured using udev rules listed in configuration files located in the **/etc/udev/rules.d** and the **/lib/udev/rules.d** directories. Default rules are already in place for symbolic links, but you can create rules of your own.

In addition to udev, information about removable devices like CD-ROMs and floppy disks, along with cameras and USB printers, used by applications like the desktop to dynamically interface with them, is managed by a separate service called the Hardware Abstract Layer (HAL). HAL allows a removable device like a USB printer to be recognized no matter what particular connections it may be using. For example, you can attach a USB printer in one USB port at one time and then switch it to another later.

HAL has a key impact on the **/etc/fstab** file used to manage file systems. No longer are entries maintained in the **/etc/fstab** file for removable devices like your CD-ROMs. These devices are managed directly by HAL using its set of storage callouts like **hal-system-storage-mount** to mount a device or **hal-system-storage-eject** to remove one. In effect you now have to use the HAL device information files to manage your removable file systems.

Floppy, USB, and Hard Disk Devices

The device name for your floppy drive is **fd0**; it is located in the directory **/dev**. **/dev/fd0** references your floppy drive. If you have more than one floppy drive, additional drives are represented by **fd1**, **fd2**, and so on. USB drives are set up similarly, beginning with **usb1**, like **/dev/usb1**.

All hard drives (SCSI, SATA, and IDE) use the prefix **sd**. The prefix for a hard disk is followed by a letter that labels the hard drive and a number for the partition. For example, **sda2** references the second partition on the first hard drive, where the first hard drive is referenced with the letter **a**, as in **sda**. The device **sdb3** refers to the third partition on the second hard drive (**sdb**). On an SATA/IDE hard disk devices, Linux supports up to four primary hard disk partitions, numbered 1 through 4. You are allowed any number of logical partitions. You can use `df` to display your hard disk partitions, which will list their device names.

RAID devices use the prefix **md**. RAID devices numbered from 0, like floppy drives. Device **md0** references the first RAID device, and **md1** references the second.

Note: Linux now manages all removable media directly with HAL, instead of using fstab entries.

DVD/CD-RW/ROM Devices

Most DVD/CD-RW/ROM devices are now either SATA or SCSI CD-ROM drives. They begin with prefix **scd** and are followed by a distinguishing number. For example, the name of a SATA/SCSI DVD/CD-RW drive could be **scd0** or **scd1**. The name of a drive attached directly to your system, was determined when you installed Linux.

DVD/CD-RW/ROM devices are configured by HAL. HAL does this in a device information file in its policy configuration directory. To configure a DVD/CD-RW device, as by adding user mount capability, you need to configure its entry in configuration file in the **/etc/udev/rules.d** directory (see Chapter 17 for details). The GNOME Volume Manager uses HAL and udev to access removable media directly. Media are mounted by **gnome-mount**, a wrapper for accessing Hal and udev, which perform the mount (**/etc/fstab** is no longer used).

Removable DVD/CD devices connected with a USB cable will have the device name of their USB connection.

Though now rarely in use, IDE DVD/CD-ROM devices are also supported. They have the old IDE prefix **hd**, and are identified by a following letter that distinguishes it from other IDE devices. For example, an IDE CD-ROM connected to your secondary IDE port may have the name **hdc**. An IDE CD-ROM connected as a slave to the secondary port may have the name **hdd**.

Mounting File Systems

Attaching a file system on a storage device to your main directory tree is called *mounting* the device. The file system is mounted to an empty directory on the main directory tree. You can then change to that directory and access those files. If the directory does not yet exist, you have to create it. The directory in the file structure to which the new file system is attached is referred to as the *mountpoint.* For example, to access files on a CD-ROM, first you have to mount the CD-ROM.

Mounting file systems can normally be done only as the root user. This is a system administration task and should not usually be performed by a regular user. As the root user, you can, however, make a particular device, like a CD-ROM, user-mountable. In this way, any user could mount a CD-ROM. You could do the same for a floppy drive.

Tip: On GNOME, you can use the Disk Management tool (**usermount**) accessible from the Applications | System Tools menu, to mount and unmount user mounted file systems, like USB drives, floppy disks, and CD-ROMs. Though risky, you could also use usermount to mount and unmount any file systems by running it as the **root** user, **sudo usermount**.

When you install your Linux system and create the Linux partition on your hard drive, your system is automatically configured to mount your main file system whenever it starts. When your system shuts down, they are automatically unmounted. You have the option of unmounting any file system, removing it from the directory tree, and possibly replacing it with another, as is the case when you replace a CD-ROM.

Once a file system it actually mounted, an entry for it is made by the operating system in the **/etc/mstab** file. Here you will find listed all file systems currently mounted.

File System Information

The file systems on each storage device are formatted to take up a specified amount of space. For example, you may have formatted your hard drive partition to take up 3GB. Files installed or created on that file system take up part of the space, while the remainder is available for new files and directories. To find out how much space you have free on a file system, you can use the **df** command or, on the desktop, either the GNOME System Monitor or the KDE KDiskFree utility. For the System Monitor, click the File Systems tab to display a list mounted file systems with information on each like the amount of the free space. KDiskFree displays a list of devices, showing how much space is free on each partition, and the percentage used.

df

The **df** command reports file system disk space usage. It lists your file systems by their device names, how much disk space they take up, and the percentage of the disk space used, as well as where they are mounted. With the **-h** option, it displays information in a more readable format; such as measuring disk space in megabytes instead of memory blocks. The **df** command is also a safe way to obtain a listing of all your partitions, instead of using **fdisk** (with **fdisk** you could erase partitions). **df** shows only mounted partitions, whereas **fdisk** shows all partitions.

```
$ df -h
Filesystem Size Used Avail Use% Mounted on
/dev/sda3  9.7G 2.8G 6.4G  31%   /
/dev/sda2  99M  6.3M 88M   7%    /boot
/dev/sda2  22G  36M  21G   1%    /home
/dev/sdc  525M  525M  0    100%  /media/disk
```

You can also use **df** to tell you to what file system a given directory belongs. Enter **df** with the directory name or **df .** for the current directory.

```
$ df .
Filesystem 1024-blocks Used Available Capacity Mounted on
/dev/sda3 297635 169499 112764 60% /
```

e2fsck and fsck

To check the consistency of the file system and repair it if it is damaged, you can use file system checking tools. **fsck** checks and repairs a Linux file system. **e2fsck** is designed to support ext2 and ext3 file systems, whereas the more generic **fsck** also works on any other file systems. The ext2 and ext3 file systems are the file systems normally used for Linux hard disk partitions and floppy disks. Linux file systems are normally **ext3**, which you would use **e2fsck** to check. **fsck** and **e2fsck** take as their argument the device name of the hard disk partition that the file system uses.

```
fsck   device-name
```

Before you check a file system, be sure that the file system is unmounted. **e2fsck** should not be used on a mounted file system. To use **e2fsck**, enter **e2fsck** and the device name that references the file system. The **-p** option automatically repairs a file system without first requesting approval from the user for each repair task. The following examples check the disk in the floppy drive and the primary hard drive:

```
e2fsck /dev/fd0
e2fsck /dev/sda1
```

With **fsck**, the **-t** option lets you specify the type of file system to check, and the **-a** option automatically repairs systems, whereas the **-r** option first asks for confirmation. The **-A** option checks all systems in the **/etc/fstab** file.

Journaling

The ext3 and ReiserFS file systems introduced journaling capabilities to Linux systems. Journaling provides for fast and effective recovery in case of disk crashes, instead of using **e2fsck** or **fsck**. With journaling, a log is kept of all file system actions, which are placed in a journal file. In the event of a sudden power loss or system crash, Linux only needs to read the journal file and replay it to restore the system to its previous (stable) state. Files that were in the process of writing to the disk can be restored to their original state. Journaling also avoids lengthy **fsck** checks on reboots that occur when your system suddenly loses power or freezes and has to be restarted physically. Instead of using **fsck** to manually check each file and directory, your system just reads its journal files to restore the file system.

Keeping a journal entails more work for a file system than a non-journal method. Though all journaling systems maintain a file system's directory structure (what is known as the *metadata*), they offer various levels of file data recovery. Maintaining file data recovery information can be time-consuming, slowing down the file system's response time. At the same time, journaling systems make more efficient use of the file system, providing a faster response time than the non-journal ext2 file system.

There are other kinds of journaling file systems you can use on Linux. These include ReiserFS, JFS, and XFS. JFS is the IBM version of a journaling file system, designed for use on servers providing high throughput such as e-business enterprise servers

(http://jfs.sourceforge.net). It is freely distributed under the GNU public license. XFS is another high-performance journaling system developed by Silicon Graphics (http://oss.sgi.com/projects/xfs). XFS is compatible with RAID and NFS file systems. Most distributions still provide support for ReiserFS file systems.

ext3 Journaling

Journaling is supported in the Linux kernel with ext3. The ext3 file system is also fully compatible with the earlier ext2 version it replaces. To create an ext3 file system, you use the **mkfs.ext3** command. Though the ext3 file system maintains full metadata recovery support (directory tree recovery), it offers various levels of file data recovery. In effect, you are trading off less file data recovery for more speed. The ext3 file system supports three options: **writeback**, **ordered**, and **journal**. The default is **writeback**. The **writeback** option provides only metadata recovery, no file data recovery. The **ordered** option supports limited file data recovery, and the **journal** option provides for full file data recovery. Any files in the process of being changed during a crash will be recovered. To specify an ext3 option, use the **data** option in the **mount** command.

```
data=ordered
```

You can upgrade ext2 file systems to ext3 versions automatically, with no loss of data or change in partitions. This upgrade just adds a journal file to an ext2 file system and enables journaling on it, using the **tune2fs** command. Be sure to change the ext2 file type to ext3 in any corresponding **/etc/fstab** entries. The following example converts the ext2 file system on **/dev/sda3** to an ext3 file system by adding a journal file (**-j**).

```
tune2fs -j /dev/sda3
```

ext4 File system

The ext4 file system enhances the ext3 file system in terms of scalability and access methods. The ext4 file system type is designed to handle very large files efficiently, supporting a much larger file size. Access methods now use extents instead of direct mapping, making access of large files much more efficient. The ext3 file system, though, remains a very effective choice for systems managing many smaller files.

Mounting File Systems Automatically: /etc/fstab

File systems are mounted using the **mount** command, described in the next section. Although you can mount a file system directly with only a **mount** command, you can simplify the process by placing mount information in the **/etc/fstab** configuration file. Using entries in this file, you can have certain file systems automatically mounted whenever your system boots. For others, you can specify configuration information, such as mountpoints and access permissions, which can be automatically used whenever you mount a file system. For example, when you add a new hard disk partition to your Linux system, you most likely want to have it automatically mounted on startup at a specific directory, and then unmounted when you shut down. Otherwise, you must mount and unmount the partition explicitly each time you boot up and shut down your system.

x

Type	Description
auto	Attempts to detect the file system type automatically.
minux	Minux file systems (filenames are limited to 30 characters).
ext4	New Linux file system format supporting long filenames and very large file sizes; includes journaling
ext3	Standard Linux file system, Includes journaling.
ext2	Older standard Linux file system. Does not have journaling.
xiaf	Xiaf file system.
msdos	File system for MS-DOS partitions (16-bit).
vfat	File system for Windows 95, 98, and for most USB drives.
reiserfs	A ReiserFS journaling file system.
xfs	A Silicon Graphics (SGI) file system.
ntfs **ntfs3g**	Windows Vista, Windows NT, Windows XP, and Windows 2000 file systems directly accessible from your Linux system.
cifs	Samba remote WIndows file systems
hpfs	File system for OS/2 high-performance partitions.
nfs	NFS file system for mounting partitions from remote systems.
nfs4	NFSv4 file system for mounting partitions from remote systems
umsdos	UMS-DOS file system.
swap	Linux swap partition or swap file.
sysv	Unix System V file systems.
iso9660	File system for mounting CD-ROM.
proc	Used by operating system for processes (kernel support file system).
sysfs	Used by operating system for devices (kernel support file system).
usbfs	Used by operating system for USB devices (kernel support file system).
devpts	Unix 98 Pseudo Terminals (ttys, kernel interface file system).
shmfs and **tmpfs**	Linux Virtual Memory, POSIX shared memory (kernel interface file system).
adfs	Apple DOS file systems.
affs	Amiga fast file systems.
ramfs	RAM-based file systems.
udf	Universal Disk Format used on CD/DVD-ROMs.
ufs	Unix File System, found on Unix system (older format).

Table 15-7: File System Types

Device Files: /dev, udev, and HAL

To mount a file system, you have to specify its device name. The interfaces to devices that may be attached to your system are provided by special files known as *device files.* The names of these device files are the device names. Device files are located in the **/dev** directories and usually have abbreviated names ending with the number of the device. For example, **fd0** may reference the first floppy drive attached to your system. The prefix **sd** references hard drives, so **sda2** would reference the second partition on first hard drive. The **sd** prefix is used for all hard drives, including IDE ATA, Serial ATA, and SCSI hard drives. In most cases, you can use the **man** command with a prefix to obtain more detailed information about this kind of device. For example, **man fd** displays the Man page for Floppy devices (you still use **man hd** for information on IDE ATA drives). A complete listing of all device names can be found in the **device-list** file at the **www.kernel.org** Web site (**www.kernel.org/pub/linux/docs/device-list/devices.txt**). Table 15-6 lists several of the commonly used device names.

Tip: The device Man pages for hard disks still use their original prefix names, even though **sd** is now used for all hard disks. The Man page **sd** displays information about SCSI hard disks and **hd** shows information about IDE hard drives.

fstab Fields

An entry in an **fstab** file contains several fields, each separated from the next by a space or tab. These are described as the device, mountpoint, file system type, options, dump, and **fsck** fields, arranged in the sequence shown here:

```
<device> <mountpoint> <filesystemtype> <options> <dump> <fsck>
```

The first field is the name of the file system to be mounted. This entry can be either a device name, a device identifier (UUID), or a file system label. A label is specified by assigning the label name to the tag **LABEL**, as in **LABEL=/** for an ext2 root partition. For hard drives, the device identifier is used (UUID), normally a serial number. A device name usually begins with **/dev**, such as **/dev/sda3** for the third hard disk partition. Though you can use a device name, the method has become deprecated and no longer advisable. The next field is the directory in your file structure where you want the file system on this device to be attached. The third field is the type of file system being mounted. Table 15-7 provides a list of all the different types you can mount. The type for a standard Linux hard disk partition is ext3. The next example shows an entry for the main Linux hard disk partition. This entry is mounted at the root directory, /, and has a file type of ext3. The device is referenced by a UUID identifier:

```
UUID=74fb43a1-0e77-47-8c4d-3ea63f87b4b    /     ext3    defaults    0   1
```

The following example shows a **LABEL** entry for the hard disk partition, where the label name is /:

```
LABEL=/     /      ext3    defaults    0   1
```

Default Fedora installations will use an LVM volume for the root partition, so you will not normally find entries like the previous ones. The /boot partitions though is a standard partition. The /boot partition will appear something like this:

```
UUID=74fb43a1-0e77-47-8c4d-3ea63f87b4b    /boot     ext3    defaults    0   1
```

mount Options

The field after the file system type lists the different options for mounting the file system. The default set of options is specified by **defaults**, and specific options are listed next to each other separated by a comma (no spaces). The **defaults** option specifies that a device is read/write (**rw**), that it is asynchronous (**async**), that it is a block device (**dev**), that it cannot be mounted by ordinary users (**nouser**), and that programs can be executed on it (**exec**).

Removable devices like your CD-ROMs and floppy disks are now managed by HAL, the Hardware Abstraction Layer. HAL uses its own configuration files to set the options for these devices. You can place your own entries in the **/etc/fstab** file for CD-ROMs to bypass HAL. This will, however, no longer let your CD-ROMs and DVD-ROMs be automatically detected.

In a HAL configuration, a DVD/CD-ROM will have **ro** and **noauto** options. **ro** specifies that the device is read-only, and **noauto** specifies it is not automatically mounted. The **noauto** option is used with both CD-ROMs and floppy drives, so they won't automatically mount, because you don't know if you have anything in them when you start up. At the same time, the HAL entries for both the CD-ROM and the floppy drive can specify where they are to be mounted when you decide to mount them. The **fscontext** option is use by SELinux as discussed in Chapter 12. Table 15-8 lists the options for mounting a file system.

Boot and Disk Check

The last two fields of an **fstab** entry consist of integer values. The first one is used by the **dump** command to determine if a file system needs to be dumped, backing up the file system. The second value is used by **fsck** to see if a file system should be checked at reboot, and in what order with other file systems. If the field has a value of 1, it indicates a boot partition, and 2 indicates other partitions. The 0 value means **fsck** needn't check the file system.

fstab Sample

A copy of an **/etc/fstab** file is shown here. Notice the first line is a comment. All comment lines begin with a **#**. The entries for the **/proc** and **/sys** file systems are special entries used by your Linux operating system for managing its processes and devices; they are not actual devices. To make an entry in the **/etc/fstab** file, you can edit the **/etc/fstab** file directly. You can use the example **/etc/fstab** file shown here as a guide to show how your entries should look.

/etc/fstab

```
# <device> <mountpoint> <filesystemtype> <options> <dump><fsck>
UUID=81acc8a8-128a-4860-bae3-999bfee5e0f5 /boot   ext3   defaults  1 2
/dev/myf10/myfed10    /          ext3    defaults        1 1
tmpfs                  /dev/shm   tmpfs   defaults        0 0
devpts                 /dev/pts   devpts  gid=5,mode=620  0 0
proc                   /proc       proc   defaults        0 0
sysfs                  /sys        sysfs  defaults        0 0
/dev/myf10/myf10swap    swap       swap   defaults        0 0
/dev/mymedia/myvideo   /mymedia    ext3   defaults        0 0
```

The boot partition is a standard ext3 partition identifies by its UUID. The other partitions, those with the swap and root system, use a LVM group called **myf10.** The root system partition is in the **myf10/myfed10** logical volume, and the swap partition is in **myf10/myf10swap** volume. In

addition, another LVM volume called **myvideo**, part of the **mymedia** LVM group (**mymedia/myvideo**), is mounted at **/mymedia**.

Option	Description
async	Indicates that all I/O to the file system should be done asynchronously.
auto	Indicates that the file system can be mounted with the **-a** option. A **mount -a** command executed when the system boots, in effect, mounts file systems automatically.
defaults	Uses default options: **rw**, **suid**, **dev**, **exec**, **auto**, **nouser**, and **async**.
dev	Interprets character or block special devices on the file system.
noauto	Indicates that the file system can only be mounted explicitly. The **-a** option does not cause the file system to be mounted.
exec	Permits execution of binaries.
nouser	Forbids an ordinary (that is, nonroot) user to mount the file system.
remount	Attempts to remount an already-mounted file system. This is commonly used to change the mount flags for a file system, especially to make a read-only file system writable.
ro	Mounts the file system as read-only.
rw	Mounts the file system as read/write.
suid	Allows set-user-identifier or set-group-identifier bits to take effect.
sync	Indicates that all I/O to the file system should be done synchronously.
user	Enables an ordinary user to mount the file system. Ordinary users always have the following options activated: **noexec**, **nosuid**, and **nodev**.
nodev	Does not interpret character or block special devices on the file system.
noexec	Does not allow execution of binaries on the mounted file systems.
nosuid	Does not allow set-user-identifier or set-group-identifier bits to take effect.

Table 15-8: Mount Options for File Systems

Partition Labels: e2label

Linux can use file system labels for ext2, ext3, and ext4 file systems on hard disk partitions. Thus in the **/etc/fstab** file previously shown, the first entry would use a label for its device name, as shown here. In this case, the label is the slash, /, indicating the root partition. You could change this device's label with **e2label**, but be sure to also change the **/etc/fstab** entry for it.

```
LABEL=/  /  ext3   defaults    0   1
```

For ext2 and ext3 partitions, you can change or add a label with the **e2label** tool or **tune2fs** with the **-L** option. Specify the device and the label name. If you change a label, be sure to change corresponding entries in the **/etc/fstab** file. Just use **e2label** with the device name to

find out what the current label is. In the next example, the user changes the label of the **/dev/sda3** device to **TURTLE**:

```
e2label /dev/sda3  TURTLE
```

Windows Partitions

You can mount MS-DOS; Windows 95/98/ME; or Windows XP, NT, and 2000 partitions used by your Windows operating system onto your Linux file structure, just as you would mount any Linux file system. You have to specify the file type of **vfat** for Windows 95/98/ME, and **msdos** for MS-DOS. Windows XP, NT, and 2000 use the **ntfs** file type. You may find it convenient to have your Windows partitions automatically mounted when you start up your Linux system. To do this, you need to put an entry for your Windows partitions in your **/etc/fstab** file and give it the **defaults** option, or be sure to include an **auto** option. You make an entry for each Windows partition you want to mount, and then specify the device name for that partition, followed by the directory in which you want to mount it. The **/mnt/windows** directory would be a logical choice (be sure the **windows** directory has already been created in **/mnt**). The next example shows a standard Windows partition entry for an **/etc/fstab** file. Notice the last entry in the **/etc/fstab** file example is an entry for mounting a Windows partition.

```
/dev/sda1 /mnt/windows vfat defaults 0 0
```

For Windows Vista, XP, NT, and 2000, you would specify the **ntfs** type. This file type is recognized by the **ntfs3g** kernel module which is automatically loaded. It detects all your NTFS drives.

```
/dev/sda2 /mnt/windows ntfs defaults 0 0
```

Linux Kernel Interfaces

Your **/etc/fstab** file may also have entries for two special kernel interface file systems, **devpts** and **tmpfs**. Both provide kernel interfaces that are not supported by standard devices. The **/dev/pts** entry mounts a **devpts** file system for pseudoterminals. The **/dev/shm** entry mounts the **tmpfs** file system (also known as **shmfs**) to implement Linux Virtual Memory, POSIX shared memory maintenance access. This is designed to overcome the 4GB memory limitation on current systems, extending usable memory to 64GB.

If your **/etc/fstab** file ever becomes corrupt—say, if a line gets deleted accidentally or changed—your system will boot into a maintenance mode, giving you read-only access to your partitions. To gain read/write access so that you can fix your **/etc/fstab** file, you have to remount your main partition. The following command performs such an operation:

```
mount -n -o remount,rw /
```

Mounting File Systems Manually: mount and umount

You can also mount or unmount any file system using the **mount** and **umount** commands, directly (notice that **umount** lacks an *n*). The mount operations discussed in the previous sections use the **mount** command to mount a file system. Normally, file systems can be mounted on hard disk partitions only by the root user, whereas CD-ROMs and floppies can be mounted by any user. Table 15-9 lists the different options for the **mount** command.

Mount Option	Description
-f	Fakes the mounting of a file system. Use it to check if a file system can be mounted.
-v	Verbose mode. **mount** displays descriptions of the actions it is taking. Use with **-f** to check for any problems mounting a file system, **-fv**.
-w	Mounts the file system with read/write permission.
-r	Mounts the file system with read-only permission.
-n	Mounts the file system without placing an entry for it in the **mstab** file.
-t *type*	Specifies the type of file system to be mounted. See Table 15-7 for valid file system types.
-a	Mounts all file systems listed in **/etc/fstab**.
-o *option-list*	Mounts the file system using a list of options. This is a comma-separated list of options following **-o**. See Table 15-8 for a list of the options.

Table 15-9: The mount Command

The mount Command

The **mount** command takes two arguments: the storage device through which Linux accesses the file system, and the directory in the file structure to which the new file system is attached. The *mountpoint* is the directory on your main directory tree where you want the files on the storage device attached. The *device* is a special device file that connects your system to the hardware device. The syntax for the **mount** command is as follows:

```
mount device mountpoint
```

Tip: The "automatic" mounting of file systems from **/etc/fstab** is actually implemented by executing a **mount -a** command in the **/etc/rc.d/rc.sysinit** file that is run whenever you boot. The **mount -a** command mounts any file system listed in your **/etc/fstab** file that does *not* have a **noauto** option. The **umount -a** command (which is executed when you shut down your system) unmounts the file systems in **/etc/fstab**.

As noted previously, device files are located in the **/dev** directories and usually have abbreviated names ending with the number of the device. For example, **fd0** may refer to the first floppy drive attached to your system. The following example mounts a hard disk (**sdc2**) to the **/mystuff** directory. The mountpoint directory needs to be empty. If you already have a file system mounted there, you will receive a message that another file system is already mounted there and that the directory is busy. If you mount a file system to a directory that already has files and subdirectories in it, those will be bypassed, giving you access only to the files in the mounted file system. Unmounting the file system, of course, restores access to the original directory files.

```
mount /dev/sdc2 /mystuff
```

For any partition with an entry in the **/etc/fstab** file, you can mount the partition using only the mount directory specified in its **fstab** entry; you needn't enter the device filename. The

mount command looks up the entry for the partition in the **fstab** file, using the directory to identify the entry and, in that way, find the device name. For example, to mount the **/dev/sda1** Windows partition in the preceding example, the **mount** command only needs to know the directory it is mounted to—in this case, **/mnt/windows**.

```
mount /mnt/windows
```

If you are unsure as to the type of file system that a disk holds, you can mount it specifying the **auto** file system type with the **-t** option. Given the **auto** file system type, **mount** attempts to detect the type of file system on the disk automatically. This is useful if you are manually mounting a floppy disk whose file system type you are unsure of (HAL would also automatically detect the file system type of any removable media, including floppies).

```
mount -t auto /dev/fd0 /media/floppy
```

The umount Command

If you want to replace one mounted file system with another, you must first explicitly unmount the one already mounted. Say you have mounted a floppy disk, and now you want to take it out and put in a new one. You must unmount that floppy disk before you can put in and mount the new one. You unmount a file system with the **umount** command. The **umount** command can take as its argument either a device name or the directory where it was mounted. Here is the syntax:

```
umount device-or-mountpoint
```

The following example unmounts the floppy disk wherever it is mounted:

```
umount /dev/fd0
```

Using the example where the device was mounted on the **/mydir** directory, you could use that directory to unmount the file system:

```
umount /mydir
```

One important constraint applies to the **umount** command. You can never unmount a file system in which you are currently working. If you change to a directory within a file system that you then try to unmount, you receive an error message stating that the file system is busy. For example, suppose a CD-ROM is mounted on the **/media/disk** directory, and then change to that **/media/disk** directory. If you decide to change CD-ROMs, you first have to unmount the current one with the **umount** command. This will fail because you are currently in the directory in which it is mounted. You have to leave that directory before you can unmount the CD-ROM.

```
# mount /dev/sdc /media/disk
# cd /media/disk
# umount /media/disk
umount: /dev/sdc: device is busy
# cd /root
# umount /media/disk
```

Tip: If other users are using a file system you are trying to unmount, you can use the **lsof** or **fuser** command to find out who they are.

Mounting Floppy Disks

As noted previously, to access a file on a floppy disk, the disk first has to be mounted on your Linux system. The device name for your floppy drive is **fd0**, and it is located in the directory **/dev**. Entering **/dev/fd0** references your floppy drive. Notice the number **0** after **fd**. If you have more than one floppy drive, the additional drives are represented by **fd1**, **fd2**, and so on. You can mount to any directory you want. Some distributions create a convenient directory to use for floppy disks, **/media/floppy**. The following example mounts the floppy disk in your floppy drive to the **/media/floppy** directory:

```
mount /dev/fd0 /media/floppy
```

Tip: On GNOME, you can mount a floppy drive by right-clicking the desktop background to display the desktop menu and then selecting Floppy in the Disk entry. To unmount, right-click the Floppy icon and select Eject from the pop-up menu.

Remember, you are mounting a particular floppy disk, not the floppy drive. You cannot simply remove the floppy disk and put in another one. The **mount** command has attached those files to your main directory tree, and your system expects to find those files on a floppy disk in your floppy drive. If you take out the disk and put another one in, you get an error message when you try to access it.

To change disks, you must first unmount the floppy disk already in your disk drive. Then, after putting in the new disk, you must explicitly mount that new disk. To do this, use the **umount** command.

```
umount /dev/fd0
```

For the **umount** or **mount** operations, you can specify either the directory it is mounted on or the **/dev/fd0** device.

```
umount /media/floppy
```

You can now remove the floppy disk, put in the new one, and then mount it:

```
mount /media/floppy
```

When you shut down your system, any disk you have mounted is automatically unmounted. You do not have to unmount it explicitly.

Mounting CD-ROMs

Remember, when you mount a CD-ROM or floppy disk, you cannot then simply remove it to put another one in the drive. You first have to unmount it, detaching the file system from the overall directory tree. In fact, the CD-ROM drive remains locked until you unmount it. Once you unmount a CD-ROM, you can then take it out and put in another one, which you then must mount before you can access it. When changing several CD-ROMs or floppy disks, you are continually mounting and unmounting them. For a CD-ROM, instead of using the **umount** command, you can use the **eject** command with the device name or mountpoint, which will unmount and then eject the CD-ROM from the drive.

To mount a DVD/CD-ROM, all you have to do is insert it into the drive. HAL will detect it and mount it automatically in the **/media** directory using the DVD/CD's label.

If instead, you want to manually mount the drive from the command line with the **mount** command, you will have to first decide on a directory to mount it to (create it if it does not exist). To manually mount a disk, use the **mount** command, the device name, like **/dev/cdrom**, and the directory it is mounted to.

```
mount /dev/cdrom  /media/cdrom1
```

If you want to manually unmount the drive, say from the command line, you can use the **unmount** command and the name of the directory it is mounted on. This may be the label of the DVD/CD-ROM.

```
umount /media/cdrom1
```

Tip: On GNOME, CD-ROMs are automatically mounted. To unmount, right-click the CD-ROM icon and select Eject from the pop-up menu.

When you burn a CD, you may need to create a CD image file. You could access such an image file from your hard drive, mounting it as if it were another file system (even ripped images could be mounted in this way). You can use the **loop** option, specifying an open loop device such as **/dev/loop0**. If no loop device is indicated, **mount** will try to find a open one. The file system type is **iso9660**, a CD-ROM ISO image file type.

```
mount -t iso9660 -o loop=/dev/loop0 image-file mount-directory
```

To mount the image file **mymusic.cdimage** to the **/mnt/mystuff** directory and make it read-only, you would use

```
mount -t iso9660 -o ro,loop=/dev/loop0 mymusic.cdimage /mnt/mystuff
```

Once it is mounted, you can access files on the CD-ROM as you would in any directory.

A disk image can also be accessed as an archive, using Archive Mounter. Right-click on the image file and select Archive Mounter. An icon for the image will be display on the desktop. You can then open the image file and browse it as an archive, listing its contents. You can even extract files from the image, copying them to another folder.

Creating File Systems: mkfs, mke2fs, mkswap, parted, and fdisk

Linux provides a variety of tools for creating and managing file systems, letting you add new hard disk partitions, create CD images, and format floppies. To use a new hard drive, you will have to first partition it and then create a file system on it. You can use either **parted** or **fdisk** to partition your hard drive. To create the file system on the partitions, you use the **mkfs** command, which is a front end for various file system builders. For swap partitions, you use a special tool, **mkswap**, and to create file systems on a CD-ROM, you use the **mkisofs** tool. Linux partition and file system tools are listed in Table 15-10.

fdisk

To start **fdisk**, enter **fdisk** on the command line with the device name of the hard disk you are partitioning. This brings up an interactive program you can use to create your Linux partition. Be careful using Linux **fdisk**. It can literally erase entire hard disk partitions and all the

data on those partitions if you are not careful. The following command invokes **fdisk** for creating partitions on the **sdb** hard drive.

```
fdisk   /dev/sdb
```

Tool	Description
fdisk	Menu-driven program to create and delete partitions.
cfdisk	Screen-based interface for **fdisk**.
parted	GNU partition management tool.
mkfs	Creates a file system on a partition or floppy disk using the specified file system type. Front end to formatting utilities.
mke2fs	Creates an ext2 file system on a Linux partition; use the **-j** option to create an ext3 file system.
mkfs.ext3	Creates an ext3 file system on a Linux partition.
mkfs.ext2	Creates an ext2 file system on a Linux partition.
mkfs.reiserfs	Creates a Reiser journaling file system on a Linux partition (links to **mkreiserfs**).
mkfs.jfs	Creates a JFS journaling file system on a Linux partition.
mkfs.xfs	Creates a XFS journaling file system on a Linux partition.
mkfs.dos	Creates a DOS file system on a given partition.
mkfs.vfat	Creates a Windows 16-bit file system on a given partition (Windows 95, 98, and ME).
mkfs.cramfs	Creates a CRAMFS compressed flash memory file system, read only (used for embedded devices).
mkswap	Tool to set up a Linux swap area on a device or in a file.
mkdosfs	Creates an MS-DOS file system under Linux.
mkisofs	Creates an ISO CD-ROM disk image.
dumpe2fs	Displays lower-level block information for a file system.
gfloppy	GNOME tool to format a floppy disk (Floppy Formatter entry on the System Tools menu).
resize2fs	Tool to extend the size of a partition, using unused space currently available on a disk.
tune2fs	Tunes a file system, setting features such as the label, journaling, and reserved block space.

Table 15-10: Linux Partition and File System Creation Tools

The partitions have different types that you need to specify. Linux **fdisk** is a line-oriented program. It has a set of one-character commands that you simply press. Then you may be prompted to type in certain information and press ENTER. If you run into trouble during the **fdisk** procedure,

you can press the **q** key at any time, and you will return to the previous screen without any changes having been made. No changes are actually made to your hard disk until you press **w**. This should be your very last command; it makes the actual changes to your hard disk and then quits **fdisk**, returning you to the installation program. Table 15-11 lists the commonly used **fdisk** commands. Perform the following steps to create a Linux partition.

When you press **n** to define a new partition, you will be asked if it is a primary partition. Press **p** to indicate that it is a primary partition. Linux supports up to four primary partitions. Enter the partition number for the partition you are creating. Enter the beginning cylinder for the partition. This is the first number in parentheses at the end of the prompt. You are then prompted to enter the last cylinder number. You can either enter the last cylinder you want for this partition or enter a size. You can enter the size as **+1000M** for 1GB, preceding the amount with a + sign. Bear in mind that the size cannot exceed your free space. You then specify the partition type. The default type for a Linux partition is 83. If you are creating a different type of partition, such as a swap partition, press T to indicate that the type you want. Enter the partition number, such as 82 for a swap partition. When you are finished, press W to write out the changes to the hard disk, and then press ENTER to continue.

Command	Action
a	Toggle a bootable flag
l	List known partition types
m	List commands
n	Add a new partition
p	Print the partition table
q	Quit without saving changes
t	Change a partition's system ID
w	Write table to disk and exit

Table 15-11: Commonly Used fdisk Commands

parted

As an alternative to **fdisk**, you can use **parted** (www.gnu.org/software/parted). **parted** lets you manage hard disk partitions, create new ones, and delete old ones. Unlike **fdisk**, it also lets you resize partitions. For you to use **parted** on the partitions in a given hard drive, none of the partitions on that drive can be in use. This means that if you wish to use **parted** on partitions located on that same hard drive as your kernel, you have to boot your system in rescue mode and choose not to mount your system files. For any other hard drives, you only need to unmount their partitions and turn your swap space off with the **swapoff** command. You can then start **parted** with the **parted** command and the device name of the hard disk you want to work on. The following example starts **parted** for the hard disk **/dev/sda**.

```
parted  /dev/sda
```

You use the **print** command to list all your partitions. The partition number for each partition will be listed in the first column under the Minor heading. The Start and End columns list the beginning and end positions that the partition uses on the hard drive. The numbers are in megabytes, starting from the first megabyte to the total available. To create a new partition, use the **mkpart** command with either **primary** or **extended**, the file system type, and the beginning and end positions. You can create up to three primary partitions and one extended partition (or four primary partitions if there is no extended partition). The extended partition can, in turn, have several logical partitions. Once you have created the partition, you can later use **mkfs** to format it with a file system. To remove a partition, use the **rm** command and the partition number. To resize a partition, use the **resize** command with the partition number and the beginning and end positions. You can even move a partition using the **move** command. The **help** command lists all commands.

mkfs

Once you create your partition, you have to create a file system on it. To do this, use the **mkfs** command to build the Linux file system and pass the name of the hard disk partition as a parameter. You must specify its full pathname with the **mkfs** command. Table 15-12 lists the options for the **mkfs** command. For example, the second partition on the first hard drive has the device name **/dev/sdb1**. You can now mount your new hard disk partition, attaching it to your file structure. The next example formats that partition:

```
mkfs -t ext3 /dev/sdb1
```

Option	Description
Blocks	Number of blocks for the file system.
-t *file-system-type*	Specifies the type of file system to format. The default is the standard Linux file system type, ext3.
file-system-options	Options for the type of file system specified. Listed before the device name, but after the file system type.
-V	Verbose mode. Displays description of each action **mkfs** takes.
-v	Instructs the file system builder program that **mkfs** invokes to show actions it takes.
-c	Checks a partition for bad blocks before formatting it (may take some time).
-l *filename*	Reads a list of bad blocks.

Table 15-12: The mkfs Options

The **mkfs** command is really just a front end for several different file system builders. A *file system builder* performs the actual task of creating a file system. Linux supports various file system builders, including several journaling file systems and Windows file systems. The name of a file system builder has the prefix **mkfs** and a suffix for the name of the type of file system. For example, the file system builder for the ext3 file system is **mkfs.ext3**. For Reiser file systems, it is **mkfs.reiserfs**, and for Windows 16-bit file systems (95, 98, ME, USB), it is **mkfs.vfat**. Some of these file builders are just other names for traditional file system creation tools. For example, the **mkfs.ext2** file builder it just another name for the **mke2fs** ext2 file system creation tool, and

mkfs.msdos is the **mkdosfs** command. As ext3 is an extension of ext2, **mkfs.ext3** simply invokes **mke2fs**, the tool for creating ext2 and ext3 file systems, and directs it to create an ext3 file system (using the **-j** option). Any of the file builders can be used directly to create a file system of that type. Options are listed before the device name. The next example is equivalent to the preceding one, creating an ext3 file system on the **sdb1** device:

```
mkfs.ext3 /dev/sdb1
```

The syntax for the **mkfs** command is as follows. You can add options for a particular file system after the type and before the device. The block size is used for file builders that do not detect the disk size.

```
mkfs options [-t type] file-sysoptions device size
```

Tip: Once you have formatted your disk, you can label it with the **e2label** command as described earlier in the chapter.

The same procedure works for floppy disks. In this case, the **mkfs** command takes as its argument the device name. It uses the ext2 file system (the default for **mkfs**), because a floppy is too small to support a journaling file system.

```
mkfs /dev/fd0
```

mkswap

If you want to create a swap partition, you first use **fdisk** or **parted** to create the partition, if it does not already exist, and then you use the **mkswap** command to format it as a swap partition. **mkswap** formats the entire partition unless otherwise instructed. It takes as its argument the device name for the swap partition.

```
mkswap /dev/sdb2
```

You then need to create an entry for it in the **/etc/fstab** file so that it will be automatically mounted when your system boots.

CD-ROM and DVD ROM Recording

Recording data to CD-ROM discs on Linux involves creating a CD image file of the CD-ROM, and then writing that image file to a CD-R or CD-RW disc in your CD-R/RW drive. With the **mkisofs** command, you can create a CD image file, which you can then write to a CD-R/RW write device. Once you create your CD image file, you can write it to a CD-write device, using the **cdrecord** or **cdwrite** application. The **cdrecord** application is a more powerful application with many options. You can also use GNOME and KDE CD recording applications such as KOnCD and GNOME Toaster to create your CDs easily. Most are front ends to the **mkisofs** and **cdrecord** tools. To record DVD discs on DVD writers, you can use **cdrecord** for DVD-R/RW drives and the dvd+rw tools for DVD+RW/R drives. If you want to record CD-ROMs on a DVD writer, you can just use **cdrecord**.

The **cdrecord** application currently works only on DVD-R/RW drives; it is part of the dvdrtools package. If you want to use DVD+RW/R drives, you would use the dvd+rw tools such as **growisofs** and **dvd+rw-format**. Some dvd+rw tools are included in the dvd+rw-tools package.

Check the DVD+RW tools Web site for more information, http://fy.chalmers.se/~appro/linux/DVD+RW.

mkisofs

To create a CD image, you first select the files you want on your CD. Then you can use **mkisofs** to create an ISO CD image of them.

You may need to include several important options with **mkisofs** to create a data CD properly. The **-o** option is used to specify the name of the CD image file. This can be any name you want to give it. The **-R** option specifies RockRidge CD protocols, and the **-J** option provides for long Windows 95/98/ME or XP names. The **-r** option, in addition to the RockRidge protocols (**-R**), sets standard global permissions for your files, such as read access for all users and no write access because the CD-ROM is read-only. The **-T** option creates translation tables for filenames for use on systems that are not RockRidge compliant. The **-U** option provides for relaxed filenames that are not standard ISO compliant, such as long filenames, those with more than one period in their name, those that begin with a period such as shell configuration files, and ones that use lowercase characters (there are also separate options for each of these features if you just want to use a few of them). Most RPM and source code package names fall in this category. The **-iso-level** option lets you remove ISO restrictions such as the length of a filename. The **-V** option sets the volume label (name) for the CD. Finally, the **-v** option displays the progress of the image creation.

The last argument is the directory that contains the files for which you want to make the CD image. For this, you can specify a directory. For example, if you are creating a CD-ROM to contain the data files in the **mydocs** directory, you would specify that directory. This top directory will not be included, just the files and subdirectories in it. You can also change to that directory and then use **.** to indicate the current directory.

If you were creating a simple CD to use on Linux, you would use **mkisofs** to first create the CD image. Here the verbose option will show the creation progress, and the **-V** option lets you specify the CD label. A CD image called **songs.iso** is created using the file located in the **newsong** directory:

```
mkisofs -v -V "Goodsongs" -o moresongs.iso  newsongs
```

If you also wanted to use the CD on a Windows system, you would add the **-r** (RockRidge with standard global file access) and **-J** (Joliet) options:

```
mkisofs -v -r -J -V "Goodsongs" -o moresongs.iso  newsongs
```

You need to include certain options if you are using filenames that are not ISO compliant, such as ones with more than 31 characters or ones that use lowercase characters. The **-U** option lets you use completely unrestricted filenames, whereas certain options like **-L** for the unrestricted length will release specific restrictions only. The following example creates a CD image called **mydoc.iso** using the files and subdirectories located in the **mdoc** directory and labels the CD image with the name "Greatdocs":

```
mkisofs -v -r -T -J -U -V "Greatdocs" -o mydocuments.iso   mydocs
```

Mounting Disk Images

Once you have created your CD image, you can check to see if it is correct by mounting it as a file system on your Linux system. In effect, to test the CD image, you mount it to a directory and then access it as if it were simply another file system.

On GNOME you can use the Archive Mounter. Right-click on the disk image file and select the Archive Mounter. The image will be mounted and accessible as a file system.

To manually mount using the **mount** command, requires the use of a loop device. Specify the loop device with the **loop** option as shown in the next example. Here the **mydoc.iso** is mounted to the **/media/cdrom** directory as a file system of type **iso9660**. Be sure to unmount it when you finish.

```
mount -t iso9660 -o ro,loop=/dev/loop0 mydocuments.iso /media/cdrom
```

Bootable CD-ROMs

If you are creating a bootable CD-ROM, you need to indicate the boot image file to use and the boot catalog. With the **-c** option, you specify the boot catalog. With the **-b** option, you specify the boot image. The *boot image* is a boot disk image, like that used to start up an installation procedure. For example, on the Fedora CD-ROM, the boot image is **isolinux/isolinux.bin**, and the boot catalog is **isolinux/boot.cat** (you can also use **images/boot.img**. and **boot.cat**). Copy those files to your hard disk. The following example creates a bootable CD-ROM image using Fedora distribution files located on the DVD/CD-ROM drive.

```
mkisofs -o fed10-0.iso -b isolinux/isolinux.bin -c isolinux/boot.cat \
  -no-emul-boot -boot-load-size 4 -boot-info-table \
 -v -r -R -T -J -V "Fed10"  /media/cdrom
```

cdrecord

Once **mkisofs** has created the CD image file, you can use Nautilus (the GNOME file manager) to directly burn an ISO image to a CD/DVD write disk. On the command line interface, you could, instead, use **cdrecord** to write it to a CD/DVD write disc. There is a command called **dvdrcecord**, but this is just a script that calls **cdrecord**, which now writes to both DVD and CD media. If you have more than one CD-writer device, you should specify the DVD/CD-R/RW drive to use by indicating its device name. In this example, the device is an IDE CD-R located at **/dev/sdc**. The **dev=** option is used to indicate this drive. The final argument for **cdrecord** is the name of the CD image file.

```
cdrecord  dev=/dev/sdc  mydocuments.iso
```

In this example, a SCSI rewritable CD-RW device with the device **/dev/scd0** is used.

```
cdrecord  dev=/dev/scd0  mydocuments.iso
```

If you are creating an audio CD, use the **-audio** option, as shown here. This option uses the CD-DA audio format:

```
cdrecord  dev=/dev/sdc -audio moresongs.iso
```

Tip: The **dummy** option for **cdrecord** lets you test the CD writing operation for a given image.

dvd+rw Tools

The primary dvd+rw tool is **growisofs**, with which you create DVD+RW/R disks. Two other minor supporting tools are also included, a formatter, dvd+rw-format, and a compatibility tool, dvd+rw-booktype. See the dvd+rw-tools page in **/usr/share/doc** for detailed instructions.

The **growisofs** tool functions like the **mkisofs** tool, except that it writes directly to the DVD+RW/R disc, rather than to an image. It has the same options as **mkisofs**, with a few exceptions, and is actually a front end to the **mkisofs** command. There is, of course, no **-o** option for specifying a disk image. You specify the DVD device instead. For example, to write the contents of the **newsongs** directory to a DVD+RW disc, you would use **growisofs** directly.

```
growisofs -v -V "Goodsongs" -Z /dev/sdc  newsongs
```

The device is specified by its name, usually **/dev/scd0** for the first SCSII device or **/dev/sdc** for the first secondary IDE drive. Recall that IDE DVD writers are configured as SCSI devices when your system boots up. **growisofs** provides a special **-Z** option for burning an initial session. For multi-sessions (DVD-RW), you can use the **mkisofs -M** option. If you want to reuse a DVD-RW disc, just overwrite it. You do not have to reformat it.

To burn an ISO image file to the disc, use the **-Z** option and assign the ISO image to the device.

```
growisofs -v -V "Goodsongs" -Z /dev/sdc=moresongs.iso
```

Though **growisofs** will automatically format new DVD+RW discs, the dvd+rw tools also include the dvd+rw-format tool for explicitly performing formats only. You use dvd+rw-format tool only to explicitly format new DVD+RW (read/write) discs, preparing them for writing. This is done only once, and only for DVD+RW discs that have never been used before. DVD+R discs do not need any formatting.

The dvd+rw-booktype tool sets the compatibility setting for older DVD-ROM readers that may not be able to read DVD+RW/R discs.

Mono and .NET Support

With Mono, Linux now provides .NET support, along with .NET applications like the Beagle desktop search tool and the F-Spot photo management tool. Mono provides an open source development environment for .NET applications. The Mono project is an open source project supported by Novell that implements a .NET framework on Unix, Linux, and OS X systems. Currently Mono 1.2 and 2.0 are included. .1.2 corresponds generally with .NET 1.1 features, and 2.0 with .NET 2.0. See **www.mono-project.com** for detailed information.

Mono is implemented on Linux using several components. These include the basic Mono .NET application, including Mono tools like the Mono certification manager (certmgr), the Global Assemblies Cache Manager tool (**gacutil**) for making assemblies available at runtime, and mcs, the Mono C# compiler. There are several additional tools for distinct features like visual basic support, SQL database queries, and .NET Web support. The Mono language testing tool, NUnit, is also included.

Configuration is found in the **/etc/mono/config** file, which is an XML-like file that maps DLL references to Linux libraries. The **/etc/mono** file also contains configuration files for 1.0 and

2.0 versions of Mono. Mono is installed in **/usr/lib/mono**. In the corresponding 1.0 and 2.0 directories you will find the DLL and EXE .NET support assemblies for different Mono applications. Other directories will hold .NET DLLs and configuration for different applications and services, including evolution, dbus, and gtk.

Local configuration information and runtime applications are placed in the user's **.config** directory.

fedora

16. LVM and RAID

With onset of cheap, efficient, and very large hard drives, even simple home systems can employ several hard drives. The use of multiple hard drives opens up opportunities for ensuring storage reliability as well as more easily organizing access to your hard disks. Linux provides two methods for better managing your hard disks: Logical Volume Manager (LVM) and Redundant Arrays of Independent Disks (RAID). LVM is a method for organizing all your hard disks into logical volumes, letting you pool the storage capabilities of several hard disks into an single logical volume. Your system then sees one large storage device, and you do not have to micromanage each underlying hard disk and its partitions. LVM is perhaps the most effective way to add hard drives to your system, creating a large accessible pool of storage. RAID is a way of storing the data on multiple hard disks. These multiple hard drives are treated as a single hard drive. They include recovery information that allows you to restore your files should one of the drives fail. The two methods can be mixed, implementing LVM volumes on RAID arrays. LVM provides flexibility and RAID can provide data protection.

With LVM you no longer have to keep track of separate disks and their partitions, trying to remember where files are stored on what partitions located in what drive. Partitions and drives and their drives are combined into logical file systems that you can attach to your system directory tree. You can have several logical file systems, each with their own drives and partitions. One major advantage to LVM file systems is that it is very easy to add new drives, expanding your memory, as well as replace older ones. The LVM file system remains intact with all its original files.

In a system with several hard drives, using both LVM and RAID, you could combined the hard drives into one logical file system, accesses the storage as one large pool. Files are stored in directories, not on directories on a particular partition. Instead of mounting file systems for each individual hard drive, there would be only one file system to mount for all the hard drives. LVM also lets you implement several logical file systems on different partitions across several hard drives.

RAID is best suited for desktops and servers which would hold multiple hard drives and require data recovery. The most favored form of RAID, RAID 5, requires a minimum of 3 hard drives. RAID, with the exception of RAID 0, provides the best protection against hard drive failure and is considered a necessity for storage intensive tasks like Enterprise, database, and Internet server operations. It can also provide peace of mind for smaller operations, providing recovery from hard disk failure. Keep in mind that there are different forms of RAID, each with advantages and weaknesses. RAID 0 provides no recovery capabilities at all. After setting up a RAID array, you could then create LVM volumes based on the array.

In comparison to LVM, RAID can provide faster access for applications that work with very large files, like multimedia, database, or graphics applications. But for normal operations, LVM is just as efficient as RAID. LVM, though, requires running your Linux system and configuring it from your Linux operating system. RAID, which is now supported at the hardware level on most computers, is easier to set up, especially a simple RAID 0 operation merely combines hard drives into one drive.

Logical Volume Manager

For easier hard disk storage management, you can set up your system to use the Logical Volume Manager (LVM), creating LVM partitions that are organized into logical volumes to which free space is automatically allocated. Logical volumes provide a more flexible and powerful way of dealing with disk storage, organizing physical partitions into logical volumes in which you can easily manage disk space. Disk storage for a logical volume is treated as one pool of memory, though the volume may in fact contain several hard disk partitions spread across different hard disks. Adding a new LVM partition merely increases the pool of storage accessible to the entire system. The original LVM package was developed for kernel 2.4. The current LVM2 package is used for kernel 2.6. Check the LVM HOWTO at www.tldp.org for detailed examples.

LVM Structure

In an LVM structure, LVM physical partitions, also known as *extents,* are organized into logical groups, which are in turn used by logical volumes. In effect, you are dealing with three different levels of organization. At the lowest level, you have physical volumes. These are physical hard disk partitions that you create with partition creation tools such as **`parted`** or **`fdisk`**. The partition type will be a Linux LVM partition, **fdisk** code **8e**. These physical volumes are organized into logical groups, known as volume groups that operate much like logical hard disks. You assign collections of physical volumes to different logical groups.

Once you have your logical groups, you can then create logical volumes. Logical volumes function much like hard disk partitions on a standard setup. For example, on the **turtle** group volume, you could create a **/var** logical volume, and on the **rabbit** logical group, you could create **/home** and **/projects** logical volumes. You can have several logical volumes on one logical group, just as you can have several partitions on one hard disk.

You treat the logical volumes as you would any ordinary hard disk partition. You create a file system on it with the **`mkfs`** command, and then you can mount the file system to use it with the **`mount`** command. For Fedora the file system type would be **ext3**.

Storage on logical volumes is managed using what are known as extents. A logical group defines a standard size for an extent, say 4MB, and then divides each physical volume in its group into extents of that size. Logical volumes are, in turn, divided into extents of the same size, which are then mapped to those on the physical volumes.

Logical volumes can be linear, striped, or mirrored. The mirror option will create a mirror copy of a logical volume, providing a restore capability. The striped option lets you automatically distribute your Logical volume across several partitions as you would a RAID device. This adds better efficiency for very large files, but is complicated to implement. Like a RAID device, stripe sizes have to be consistent across partitions. As LVM partitions can be of any size, the stripes sizes have to be carefully calculated. The simplest approach is just to use a linear implementation, much like a RAID 0 device, just treating the storage as one large ordinary drive, with storage accessed sequentially.

There is one restriction and recommendation for logical volumes. The boot partition cannot be part of a logical volume. You still have to create a separate hard disk partition as your boot partition with the **/boot** mountpoint in which your kernel and all needed boot files are installed. In addition, it is recommended that you not place your root partition on a logical volume.

Doing so can complicate any needed data recovery. This is why a default partition configuration set up during Fedora installation will include a separate **/boot** partition of 100MB of type **ext3**, whereas the root and swap partitions will be installed on Logical volumes. There will be two partitions, one for the logical group (LVM physical volume, **pv**) holding both swap and root volumes, and another for the boot partition (**ext3**). The logical volumes will in turn both be **ext3** file systems.

Creating LVMs during Installation

Creating logical volumes involves several steps. First, you create physical LVM partitions, then the volume groups you place these partitions in, and then from the volume groups you create the logical volumes for which you then specify mountpoints and file system types. You can create LVM partitions during the installation process when the partition screen is displayed. First you create your physical LVM partitions. On the partition screen, click the New button and select "physical volume (LVM)" for the File System Type. Often you will be using an entire hard disk for an LVM group. In this case you can create a single LVM physical partition for the entire hard disk. If you wish, you could create several LVM physical partitions on your hard disks. If you want more than one Volume group, each will have to have at least one LVM physical partition.

Once you have created LVM physical partitions, you click the LVM button to create your logical groups and volumes. You first need to assign the LVM physical partitions to a volume group. Volume groups are essentially logical hard drives. If you volume group uses several hard drives, you could assign LVM physical partitions from the different hard drives to the same volume group, letting the volume group span different hard drives.

Once the volume groups are created, you are ready to create your logical volumes. You can create several logical volumes within each group. The logical volumes function like partitions. You will have to specify a file system type and mountpoint for each logical volume you create. The file system type for Fedora is normally **ext3**.

system-config-lvm

The system-config-lvm tool provides a GUI interface for managing your Logical Volume Manager. With it you can obtain information about your logical and physical volumes, as well as perform simple tasks such as deleting and extending logical volumes, or migrating and removing physical volumes. You can invoke system-config-lvm from its menu entry in the System | Administration | Logical Volume Management. You can also enter **system-config-lvm** in a terminal window.

system-config-lvm will display a window with three panes: one listing all your logical and physical volumes, one showing a graphical representation of a selected volume or volume group, and one that display information about the selected volumes. Figure 16-1 shows two volume groups. **VolGroup00** is the default volume group set up during installation, whereas mymedia is one set up later by the user. **VolGroup00** in implemented on a physical volume, a pv partition, which is the third partition on the first hard drive, **sda3**. The two logical volumes for the group are used for the root and swap portions, **LogVol01** for the root partition, and **LogVol00** for the swap partition.

Selecting a physical volume displays buttons with the options to extend the volume group or remove physical volumes, whereas selecting a particular partition allows you to migrate a

particular volume or remove it from the group. When extending a volume group you will be presented with a list of possible partitions to choose from.

Selecting a logical group shows buttons to create or remove the volume, and selecting a particular volume in that group permits you to remove the logical volume or edit its properties. A logical volume's properties will let you specify its file system type, size, and logical volume name. When adding a new logical volume you can use properties to set the name, size, and file system type, formatting it appropriately. Space permitting you can even resize current volumes.

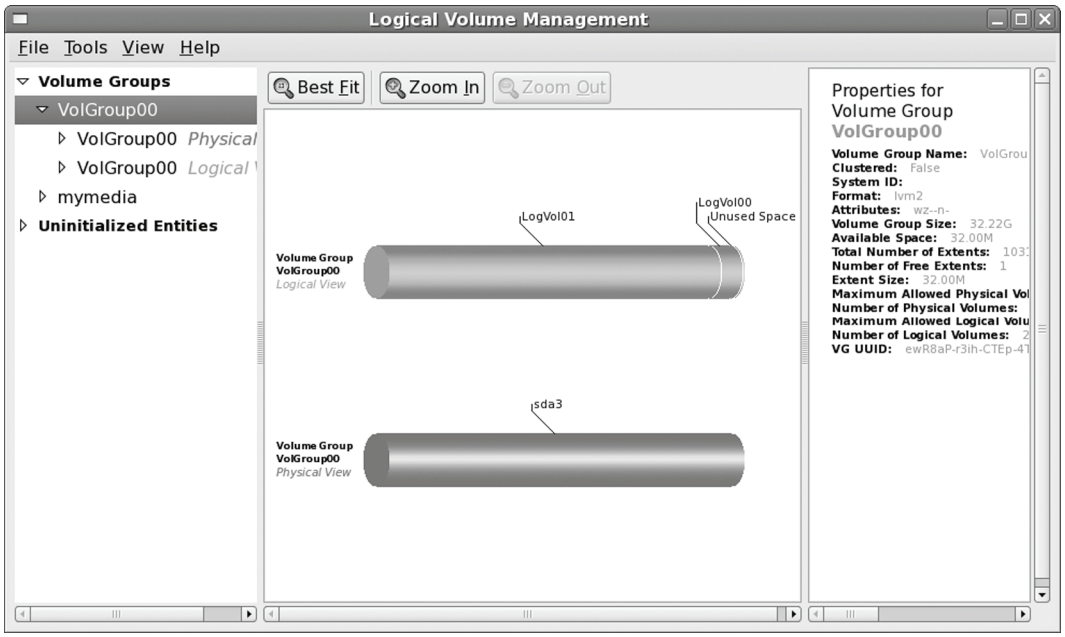


Figure 16-1: GUI Logical Volume Manager administration (system-config-lvm)

The uninitialized entries are partitions that do not belong to any volume. Recall, though, that the boot partition cannot belong to a volume group, it cannot be a logical volume. Be sure to leave it alone. Also, if you have and Windows (NTFS) or other Linux partitions (ext3 that your are using, be sure to leave them alone. For other uninitialized partitions that you may want to add to your LVM group, you can select their entries and initialize them to add them to the volume group. Use the Initialize Block Device entry in the Tools menu.

If you have the free space on a logical group you can create a new logical volume. First select the logical group entry on the left hand pane. Then Click on the Create New Logical Volume that will then appear (see Figure 16-4). This opens up a new window with a panel for creating a new Logical volume (see Figure 16-2). There are entries for the volume name, its size, the file system you want it formatted with, and where you want it mounted. You also have the option of specifying the size of the extents, though the default normally works well You can specify whether a logical volume should be linear, mirrored, or striped. These features are similar to the linear, mirrored, or striped implementations used in RAID devices. Normally you would use the linear implementation which is the default.

To extend the size of a volume using free space in the volume group, just select the volume group and click the Edit properties button. This open the same window as displayed in Figure 16-2. You can then use the slider on the volume size to extend the size of the volume. Upon clicking OK, system-config-lvm will unmount your volume group and then resize the volume and check the file system, extending the size while preserving your original data. This capability is a major advantage for LVM devices. Hard disk partitions are fixed, whereas LVM logical disks can easily be expanded. To expand a hard disk partition you had to destroy the old one and create a new larger one that in turn was also fixed. With LVM you just add more storage. The logical structure is separated from the physical implementation.

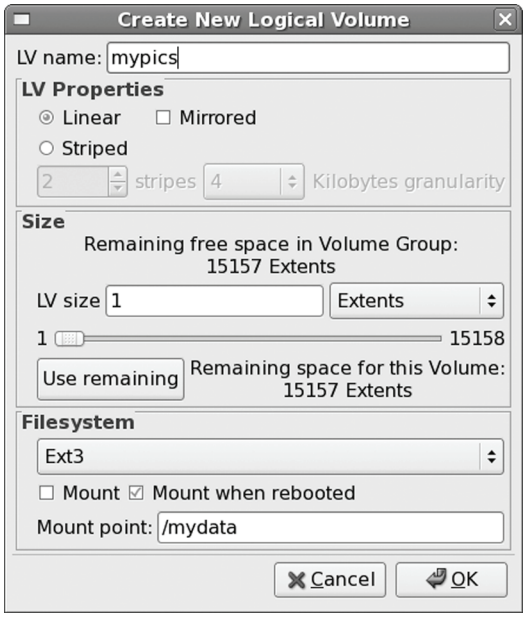


Figure 16-2: Creating a new logical volume with LVM.

LVM Tools: using the LVM commands

Instead of using system-config-lvm, you could use a collection of LVM tools to manage your LVM volumes, adding new LVM physical partitions and removing current ones. The system-config-lvm system tool is actually a GUI interface for the LVM tools. For the LVM tools, you can either use LVM tools directly or use the **`lvm`** command to generate an interactive shell from which you can run LVM commands. There are Man pages for all the LVM commands. LVM maintains configuration information in the **/etc/lvm/lvm.conf** file, where you can configure LVM options such as the log file, the configuration backup directory, or the directory for LVM devices (see the **lvm.conf** Man page for more details).

Displaying LVM Information

You can use the **`pvdisplay`**, **`vgdisplay`**, and **`lvdisplay`** commands to show detailed information about a physical partition, volume groups, and logical volumes. **`pvscan`**, **`vgscan`**, and **`lvscan`** list your physical, group, and logical volumes.

Managing LVM Physical Volumes with the LVM commands

A physical volume can be any hard disk partition or RAID device. A RAID device is seen as a single physical volume. You can create physical volumes either from a single hard disk or from partitions on a hard disk. On very large systems with many hard disks, you would more likely use an entire hard disk for each physical volume.

You would first use a partition utility like **fdisk**, **parted**, or **gparted** to create a partition of the LVM partition type (**8e**). Then you can initialize the partition as a physical volume using the **pvcreate** command.

To initialize a physical volume on an entire hard disk, you use the hard disk device name, as shown here:

```
pvcreate /dev/sdc
```

This will initialize one physical partition, **pv**, called **sdc1** on the **sdc** hard drive (the third Serial ATA drive, c).

If you are using a particular partition on a drive, you create a new physical volume using the partition's device name, as shown here:

```
pvcreate /dev/sda3
```

To initialize several drives, just list them. The following create two physical partitions, sdc1 an sdd1.

```
pvcreate /dev/sdc /dev/sdd
```

You could also use several partitions on different hard drives. This is a situation in which your hard drives each hold several partitions. This condition occurs often when you are using some partitions on your hard drive for different purposes like different operating systems, or if you want to distribute your Logical group across several hard drives. To initialize these partitions at once, you simply list them.

```
pvcreate /dev/sda3 /dev/sdb1 /dev/sdb2
```

Once you have initialized your partitions, you have to create LVM groups on them.

Managing LVM Groups

Physical LVM partitions are used to make up a volume group. You can manually create a volume group using the **vgcreate** command and the name of the group along with a list of physical partitions you want in the group.

If you are then creating a new volume group to place these in, you can include them in the group when you create the volume group with the **vgcreate** command. The volume group can use one or more physical partitions. The default install configuration described previously used only one physical partition for the **VolGroup00**. In the following example, a volume group called **mymedia** that is made up two physical volumes, **sdc** and **sdd**.

```
vgcreate mymedia  /dev/sdc /dev/sdd
```

The previous example sets up a logical group on two serial ATA hard drives, each with its own single partition. Alternatively, you can set up a volume group to span partitions on several hard drives. If you are using partitions for different functions, this approach gives you the

flexibility for using all the space available across multiple hard drives. The following example creates a group called **mygroup** consisting of three physical partitions, **/dev/sda3**, **/dev/sdb4**, and **/dev/sdb4**:

```
vgcreate mygroup  /dev/sda3 /dev/sdb2 /dev/sdb4
```

If you to later want to add a physical volume to a volume group you use the **vgextend** command. The **vgextend** command adds a new partition to a logical group. In the following example, the partition **/dev/sda4** is added to the volume group **mygroup**. In effect, you are extending the size of the logical group by adding a new physical partition.

```
vgextend mygroup  /dev/sda4
```

To add an entire new drive to a volume group you would follow a similar procedure. The following example adds a fifth serial ATA hard drive, **sde**, first creating a physical volume on it and then adding that volume, sde, to the **mymedia** volume group.

```
pvcreate /dev/sde
vgextend mymedia /dev/sde
```

To remove a physical partition, first remove it from its logical group. You may have to use the **pmove** command to move any data off the physical partition. Then use the **vgreduce** command to remove it from its logical group.

You can remove a entire volume group by first deactivating it with **vgchange -a n** and then using the **vgremove** command.

Activating Volume Groups

Whereas in a standard file system structure, you mount and unmount hard disk partitions, with an LVM structure, you activate and deactivate entire volume groups. The group volumes are accessible until you activate them with the **vgchange** command with the **-a** option. To activate a group, first reboot your system, and then enter the **vgchange** command with the **-a** option and the **y** argument to activate the logical group (an **n** argument will deactivate the group).

```
vgchange -a  y  mygroup
```

Managing LVM Logical Volumes

To create logical volumes, you use the **lvcreate** command and then format your logical volume using the standard formatting command like **mkfs.ext3**. Keep in mind that all these actions can be performed at once by system-config-lvm (see Figure 16-2).

With the **-n** option you specify the volume's name, which functions like a hard disk partition's label. You use the **-L** or **--size** options to specify the size of the volume. Use a size suffix for the measure, **G** for Gigabyte, **M** for megabyte, and **K** for kilobytes. There are other options for implementing features like whether to implement a linear, striped, or mirrored volume, or to specify the size of the extents to use. Usually the defaults work well. The following example creates a logical volume named **projects** on the **mygroup** logical group with a size of 20GB.

```
lvcreate -n projects  -L 20GB mygroup
```

The following example sets up a logical volume on the **mymedia** volume group that is 540GB in size. The mymedia volume group is made up of two physical volumes each on 320GB hard drives. In effect the two hard drives are logically seen as one.

```
lvcreate -n myvideos  -L 540GB mymedia
```

Once you have created your logical volume, you then have to create a file system to use on it. The following creates an ext3 file system on the myvideos logical volume.

```
mkfs.ext3 myvideos
```

You could also use:

```
mkfs -t ext3 myvideos
```

With **lvextend**, you can increase the size of the logical volume if there is unallocated space available in the volume group.

Should you want to reduce the size of a logical volume you use the **lvreduce** command, indicating the new size. Be sure to first reduce the size of any file systems (**ext3**) on the logical volume, using the **resize2fs** command.

To rename a logical volume use the **lvrename** command. If you want to completely remove a logical volume, you can use the **lvremove** command.

Steps to create a new LVM group and volume

- **Physical Partition** First create a physical partition on your hard drive. You can use GParted, QTparted, or fdisk with the disk device name to create the partition. For example, to use fdisk to create a new partition on a new hard drive, whose device name is **/etc/sde**, you would enter:

  ```
  fdisk /etc/sde
  ```

  Then, in the fdisk shell, use the fdisk **n** command to create a new partition, set it as a primary partition (**p**), and make it the first partition. If you plan to use the entire hard drive for your LVM, you would need only one partition that would cover the entire drive.

  Then use the **t** command to set the partition type to 8E. The 8E type is the LVM partition type. To make your changes, enter **w** to write changes to the disk.

- **Physical Volume** Next create a physical volume (pv) on the new and empty LVM partition, using the **pvcreate** command and the device name.

  ```
  pvcreate /dev/sde
  ```

- **Volume Group** Then create your volume group with **vgcreate** command, with the volume group name and the hard disk device name.

  ```
  vgcreate mynewgroup  /dev/sde
  ```

  Be sure the volume group is activated. Use the **vgs** command to list it. If not listed, use the following command to activate it.

  ```
  vgchange -a  y  mynewgroup
  ```

- **Logical Volume** Then create a logical volume or volumes for the volume group, using the **lvcreate** command. The **--size** or **-L** options determines the size and the **--name** option specifies the name. To find out the available free space use the **vgs** command. You can have more than one logical volume in a volume group, or just one if you prefer. A logical volume is conceptually similar to logical volumes in a extended partition on Windows systems.

```
lvcreate --size --name mynewvol1
```

- **Format the Logical volume**. You then use the **mkfs** command with the **-t** option to format the logical volume. On Fedora, the logical volume will be listed in a directory for the LVM group, within the /dev directory, **/dev/mynewgroup/mynewvol1**. This is a link to the **/dev/mapper** directory that actually holds the device file prefix with the volume group name, **/dev/mapper/mynewgroup-mynewvol1**.

```
mkfs -t ext3 /dev/mapper/mynewgroup-mynewvol1
```

Steps to add a new drive to a LVM group and volume

- **Physical Partition** First create a physical partition on your hard drive. You can use GParted, QTparted, or fdisk with the disk device name to create the partition. For the type specify LVM (**8E**).
- **Physical Volume** Next create a physical volume (pv) on the new and empty LVM partition, using the pvcreate command and the device name.

```
pvcreate /dev/sdf
```

- **Add to Logical Group** Use the vgextend command to add the new physical volume to your existing logical group (LG).

```
vgextend  mynewgroup  /dev/sdf
```

- **Add to Logical Volume** Then you can use then create new logical volumes in the new space, or expand the size of a current logical volume. To expand the size of a logical volume to the new space, first umount the logical volume. Then use the **lvextend** command to expand to the space on the new hard drive that is now part the same logical group. With no size specified, the entire space on the new hard drive will be added.

```
umount /dev/mynewgroup/mynewvol1
lvextend /dev/mynewgroup/mynewvol  /dev/sdf
```

  Use the **-L** option to specify a particular size, **-L +250G** . Be sure to add the + sign to have the size added to the current logical volume size. To find out the available free space, use the **vgs** command.

- **Add to file system** Then use the **resize2fs** command to extend the linux file system (ext3) on to logical volume to include the new space, formatting it. Unless you specify a size (second parameter), then all the available unformatted space is used.

```
resize2fs /dev/mynewgroup/mynewvol1
```

LVM Device Names: /dev/mapper

The **device-mapper** driver is used by LVM to set up tables for mapping logical devices to hard disk. The device name for a logical volume is kept in the **/dev/mapper** directory and has the format *logical group –logical volume*. In addition, there will be a corresponding device folder for the logical group, which will contain logical volume names. These device names are links to the **/dev/mapper** names. For example, the **mypics** logical volume in the **mymedia** logical group has the device name, **/dev/mapper/mymedia-mypics**. There will be a corresponding folder called **/dev/mymedia**. The device name **/dev/mymedia/mypics** is a link to the **/dev/mapper/mymedia-mypics** device name. You can just as easily use the link as shown in this chapter, as the original device name. The snapshot device described later in this chapter would have the device name **/dev/mapper/mymedia-mypicssnap1** with the link device name being **/dev/mymedia/mypicssnap**.

Note: You can back up volume group meta data (configuration) using the **vgcfgbackup** command. This does not backup your logical volumes (no content). Meta data backups are stored in **/etc/lvm/backup**, and can be restored using **vgcfgrestore**.

Using LVM to replace drives

LVM can be very useful when you need to replace an older hard drive with a new one. Hard drives are expected to last about six years on the average. You could want to replace the older drive with a larger one (hard drive storage sizes double every year or so). Replacing drives is easy for additional hard drives. To replace a boot drive is much more complicated.

To replace the drive, simply incorporate the new drive to your logical volume (see Steps to add a new drive to an LVM group and volume). The size of your logical volume will increase accordingly. You can use the **pmove** command to move data from the old drive to the new one. Then, issue commands to remove the old drive (**vgreduce**) from the volume group. From the user and system point of view, no changes are made. Files from your old drive will still be stored in the same directories, though the actual storage will be implemented on the new drive.

Replacement with LVM become more complicated if you want to replace your boot drive, the hard drive from which your system starts up and which holds your linux kernel. The boot drive contains a special boot partition and the master boot record. The boot partition cannot be part of any LVM volume. You would first have to create a boot partition on the new drive using a partition tool such as Parted or fdisk, labeling it as boot. The boot drive is usually very small, about 200 MB. Then mount the partition on your system, copy the contents of your **/boot** directory to it. Then you add the remainder of the disk to your logical volume and logically remove the old disk, copying the contents of the old disk to the new one. You would still have to boot with linux rescue DVD (or install DVD in rescue mode), and issue the **grub-install** command to install the master boot record on your new drive. You can then boot from the new drive.

LVM Example for multiple hard drives

With hard drives becoming cheaper and the demand for storage increasing, many systems now use multiple hard drives. To manage multiple hard drives, partitions on each had to individually managed, unless you implemented a RAID system. RAID allows you to tread several hard disks as one storage device, but there restrictions on the size and kinds of devices you can

combine. Without RAID, each hard drives had to be managed separately, with files having to fit into remaining storage as the drives filled up.

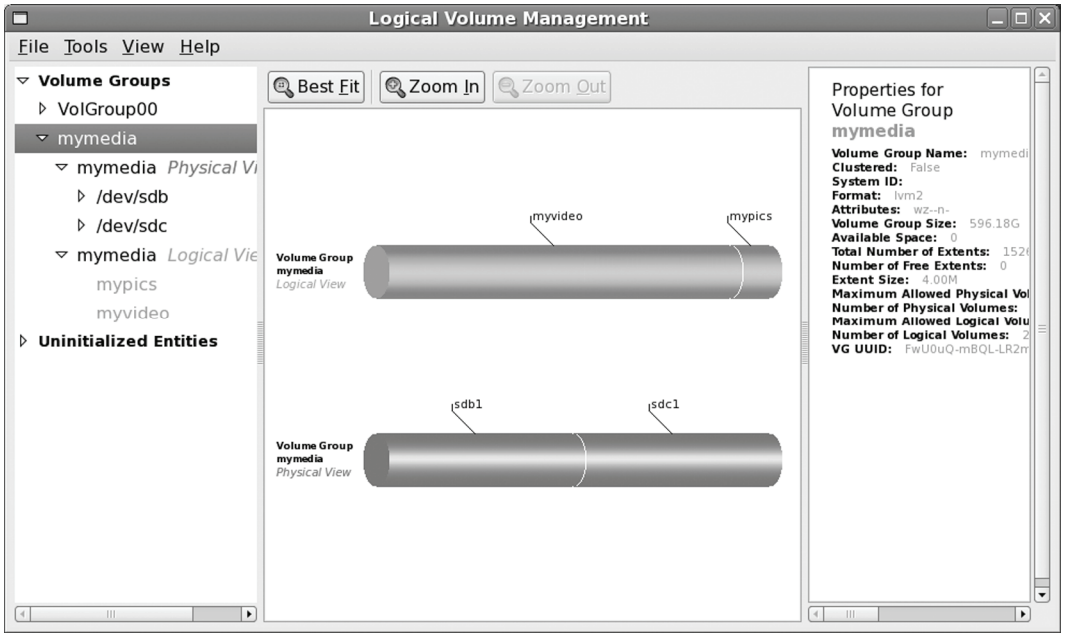


Figure 16-3: Logical Volume Management with two hard drives using system-config-lvm

With LVM, you no longer have such restrictions. You can combine hard disks into a single storage device. The device is also flexible, letting you replace disks without loosing any data, as well adding new disks to automatically increase your storage (or replace smaller disks with larger ones).

For example, say you want to add two hard disks to your system, but want to treat the storage in both logically, instead of having to manage partitions in each. LVM lets you treat the combined storage of both hard drives as one giant pool. In effect, two 500GB drives could be treated as one 1 Terabyte storage device.

In this example, the Linux system makes use of three hard drives. The Linux system and boot partitions are on the first hard drive, **sda**. Added to this system are two hard drives, **sdc** and **sdd** which will make up an LVM storage device to be added to the system.

Using system-config-lvm

Using the example in Figure 16-3, the steps involved in creating and accessing logical volumes are described in following commands. In this example there are two hard disk drives that will be combined into one LVM drives. The hard drives are Serial ATA drives identified on the systems as **sdc** and **sdd**. Each drive is first partitioned with a single LVM physical partitions. On system-config-lvm, the drives will be listed as uninitialized. Select them and initialize them to

create the physical partitions on the hard disks. In this example, the partitions **sdc** and **sdd** are created.

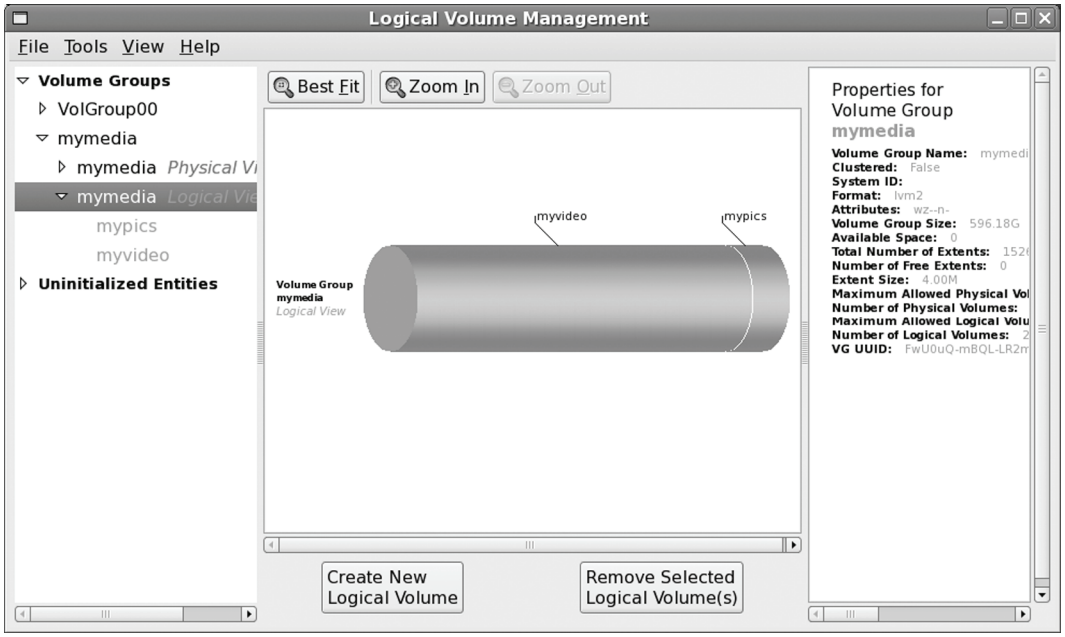


Figure 16-4: Logical volumes with system-config-lvm

You then create then volume group, and then create your logical volumes (see Figure 16-4). For logical volumes, select the logical group entry on the left hand pane. Then Click on the Create New Logical Volume. This opens up a new window with a panel for creating a new Logical volume . Figure 16-4 shows the **myvideo** and **mypics** logical volumes which are part of the mymedia logical group.

Using LVM commands

Using the LVM commands you can achieve the same effect. You first initialize the physical volumes on the **sdc** and **sdd** drives with the **`pvcreate`** command. The **sda1** and **sda2** partitions in the **sda** entry are reserved for the boot and root partitions and are never initialized.

```
pvcreate /dev/sdc /dev/sdd
```

You then create the logical groups you want, using the **`vgcreate`** command. In this case there are one logical group, **mymedia**. The **mymedia** group uses **sdc** and **sdd**. If you create a physical volume later and want to add it to a volume group, you would use the **`vgextend`** command.

```
vgcreate mymedia  /dev/sdc /dev/sdd
```

You can now create the logical volumes in each volume group, using the **`lvcreate`** command. In this example two logical volumes are created, one for **myvideos** and another for **mypics**. The corresponding **`lvcreate`** commands are shown here:

```
lvcreate  -n myvideo   -l 540GB    mymedia
lvcreate  -n mypics    -l 60GB    mymedia
```

Then you can activate the logical volumes. Reboot and use **vgchange** with the **-a y** option to activate the logical volumes.

```
vgchange -a y mymedia
```

You can now create the file systems for each logical volume.

```
mkfs.ext3 myvideo
mkfs.ext3 mypics
```

Then you can mount the logical volumes. In this example they are mounted to subdirectories of the same name in **/mydata**, **/mydate/mypics** and **/mydata/myvideo**.

```
mount -t ext3 /dev/mymedia/mypics  /mydata/mypics
mount -t ext3 /dev/mymedia/myvideo /mydata/myvideo
```

LVM Snapshots

A snapshot records and defines the state of the logical volume at a designated time. It does not create a full copy of data on the volume, just changes since the last snapshot. A snapshot defines the state of the data at a given time. This allows you to backup the data in a consistent way. Should you need to restore a file to its previous version, you can use the snapshot of it. Snapshots are treated as logical volume and can be mounted, copied, or deleted.

To create a snapshot you use the lvcreate command with the -s option. In this example the snapshot is given the name mypics-snap1 (-n option). You need to specify the full device name for the logical group you want to create the snapshot for. Be sure there is enough free space available in the logical group for the snapshot. In this example, the snapshot logical volume is created in the /dev/mymedia logical group. It could just a s easily be created in any other logical group. Though a snapshot normally uses very little space, you have to guard against overflows. If the snapshot is allocated same size as the original, it will never overflow For systems where little of the original data changes, the snapshot can be very small. The following example allocate one third the size of the original (60GB).

```
lvcreate -s –n mypics-snap1 -l 20GB /dev/mymedia
```

You can then mount the snapshot as you would any other file system.

```
mount /dev/mymedia/mypic-snap1 /mysnaps
```

To delete a snapshot you use the lvremove command, removing it like you would any logical volume.

```
lvremove -f /dev/mymedia/mypics-nap1
```

Snapshots are very useful for making backups, while a system is still active. You can sue tar or dump to backup the mounted snapshot to a disk or tape. All the data form the original logical volume will be included, along with the changes noted by the snapshot.

Snapshots also allow you to perform effective undo operations. You can create snapshot of a logical volume. Then unmount the original and mount the snapshot in its place. Any changes you make will be performed on the snapshot, not the original. Should problems occur, unmount the snapshot and then mount the original. This restores the original state of your data. You could also

do this using several snapshots, restoring to a previous snapshot. With this procedure, you could test new software on a snapshot, without endangering your original data. The software would be operating on the snapshot, not the original logical volume.

You can also use them as alternative versions of a logical volume. You can read and write to a snapshot. A write will change only the snapshot volume, not the original, creating, in effect, a alternate version.

Configuring RAID Devices

RAID is a method of storing data across several disks to provide greater performance and redundancy. In effect, you can have several hard disks treated as just one hard disk by your operating system. RAID then efficiently stores and retrieves data across all these disks, instead of having the operating system access each one as a separate file system. Lower-level details of storage and retrieval are no longer a concern of the operating system. This allows greater flexibility in adding or removing hard disks, as well as implementing redundancy in the storage system to provide greater reliability. With RAID, you can have several hard disks that are treated as one virtual disk, where some of the disks are used as real-time mirrors, duplicating data. You can use RAID in several ways, depending upon the degree of reliability you need. When you place data on multiple disks, I/O operations can overlap in a balanced way, improving performance. Because having multiple disks increases the mean time between failures (MTBF), storing data redundantly also increases fault tolerance.

RAID can be implemented on a hardware or software level. On a hardware level, you can have hard disks connected to a RAID hardware controller, usually integrated into a PC chipset (Intel, Nvidia, or AMD) or provided by a separate RAID card. Your operating system then accesses storage through the RAID hardware controller. Alternatively, you can implement RAID as a software controller, letting a software RAID controller program manage access to hard disks treated as RAID devices. Linux uses the MD driver to implement a software RAID controller. Linux software RAID supports a few levels (linear, 0, 1, 4, 5, 6 and 10), whereas hardware RAID supports many more. Hardware RAID levels, such as 7–10, provide combinations of greater performance and reliability.

Motherboard RAID Support: dmraid

With kernel 2.6, hardware RAID devices are supported with the Device-Mapper Software RAID support tool (dmraid) module, which currently supports a wide range of hardware motherboard RAID devices. Keep in mind that many “hardware” RAID devices are, in effect, really software RAID (fakeraid). Though you configure them in the motherboard BIOS, the drivers operate as software, like any other drivers. In this respect they could be considered less flexible than a Linux software RAID solution, and they could also depend directly on vendor support for any fixes for updates.

This The dmraid module driver will map your system to hardware RAID devices such as those provided by Intel, Promise, and Silicon Magic, and often included on motherboards. The dmraid tool uses the device-mapper driver to set up a virtual file system interface, just as is done for LVM drives. The RAID device names will be located in /dev/mapper.

You use your BIOS RAID configuration utility to set up your RAID devices as instructed by your hardware documentation. During a Linux installation, the RAID devices are automatically detected and the dmraid module is loaded, selecting the appropriate drivers.

With the dmraid command you can detect and activate RAID devices. The following command would displays your RAID devices:

```
dmraid -r
```

To list currently supported devices, use dmraid with the -l option.

```
dmraid -l
```

The dmraid tool is improved continually and may not work well with some RAID devices.

Linux Software RAID Levels

Linux software RAID can be implemented at different levels, depending on whether you want organization, efficiency, redundancy, or reconstruction capability. Each capability corresponds to different RAID levels. For most levels, the size of the hard disk devices should be the same. For mirroring for RAID 1, disks of the same size are required, and for RAID 5 they are recommended. Linux software RAID supports several levels, as shown in Table 16-2. RAID 5 requires at least three hard drives. In addition, FAULTY raid level is provided for testing purposes (On Red Hat Enterprise Linux and Fedora, level 4 is implemented as part of level 5.)

RAID Level	Capability	Description
Linear	Appending	Simply treats RAID hard drives as one virtual drive with no striping, mirroring, or parity reconstruction.
0	Striping	Implements disk striping across drives with no redundancy.
1	Mirroring	Implements a high level of redundancy. Each drive is treated as mirror for all data.
4	Parity	Implements data reconstruction capability using parity information, kept on a separate disk.
5	Distributed parity	Implements data reconstruction capability using parity information. Parity information is distributed across all drives, instead of using a separate drive as in RAID 4. Requires at least three hard drives.
6	Distributed parity	Implements data reconstruction capability using dual distributed parity information. Dual sets of parity information is distributed across all drives. Can be considered an enhanced form of 5.
10	Striping and Mirroring	Implements a high level of redundancy with striping. Also know as 1+0
Multipath	Multiple access to devices	Supports multiple access to the same device

Table 16-1: Linux Software RAID Levels

Note: For mirroring for RAID 1, disks of the same size are required, and for RAID 5 they are recommended.

Linear

The *linear* level lets you simply organize several hard disks into one logical hard disk, providing a pool of continuous storage. Instead of being forced to set up separate partitions on each hard drive, in effect you have only one hard drive. The storage is managed sequentially. When one hard disk fills up, the next one is used. In effect, you are *appending* one hard disk to the other. If you had a hard disk RAID array containing two 80GB disks, after you used up the storage on one, you would automatically start on the next. This level provides no recovery capability.

RAID 0: Striping

For efficiency, RAID stores data using disk *striping,* where data is organized into standardized stripes that can be stored across the RAID drives for faster access (level 0). RAID 0 also organizes your hard disks into common RAID devices but treats them like single hard disks, storing data randomly across all the disks. If you had a hard disk RAID array containing two 80GB disks, you could access them as one 160GB RAID device.

Tip: Keep in mind that many "hardware" RAID devices are, in effect, really software RAID (fakeraid). Though you configure them in the BIOS, the drivers operate as software, like any other drivers. In this respect they could be considered less flexible than a Linux software RAID solution, as well as depending directly on vendor support for any fixes for updates.

RAID 1: Mirroring

RAID level 1 implements redundancy through *mirroring.* In mirroring, the same data is written to each RAID drive. Each disk has a complete copy of all the data written, so that if one or more disks fail, the others still have your data. Though extremely safe, redundancy can be very inefficient and consumes a great deal of storage. For example, on a RAID array of two 80GB disk drives, one disk is used for standard storage and the other is a real-time backup. This leaves you with only 80GB for use on your system. Write operations also have to be duplicated across as many mirrored hard disks as are used by the RAID array, slowing down operations.

RAID 5 and 6: Distributed Parity

As an alternative to mirroring, data can be reconstructed using *parity information* in case of a hard drive crash. Parity information is saved instead of full duplication of the data. Parity information takes up the space equivalent of one drive, leaving most of the space on the RAID drives free for storage. RAID 5 combines both striping and parity (see RAID 4), where parity information is distributed across the hard drives, rather than in one drive dedicated to that purpose. This allows the use of the more efficient access method, striping. With both striping and parity, RAID 5 provides both fast access and recovery capability, making it the most popular RAID level used. For example, a RAID array of four 80GB hard drives would be treated as one 320GB hard drive with part of that storage (80 GB) used to hold parity information, giving 240GB free. RAID 5 does require at least three hard drives.

RAID 6 operates the same as RAID 5, but it uses dual sets of parity information for the data, providing even greater restoration capability.

RAID 4: Parity

Though it is not supported in Red Hat Enterprise Linux and Fedora due to overhead costs, RAID 4, like RAID 5, supports a more compressed form of recovery using parity information instead of mirrored data. With RAID 4, parity information is kept on a separate disk, while the others are used for data storage, much like in a linear model.

Tip: Red Hat Enterprise Linux and Fedora also allows you to create and format Linxu software RAID devices during installation. At that time, you can create your RAID partitions and devices.

RAID 10: Striping and Mirroring

RAID 10, also known as RAID 1+0, combines both striping and mirroring (RAID 0 and 1). This provides both redundancies and fast access.

Multipath

Though not actually a RAID level, Multipath allows for multiple access to the same device. Should one controller fail, another can be used to access the device. In effect, you have controller-level redundancy. Support is implemented on Red Hat Enterprise Linux and Fedora using the **mdadmd** daemon. This is started with the **mdadmd** service script.

```
start mdadmd start
```

RAID Devices and Partitions: md and fd

A RAID device is named an **md** and uses the MD driver. These devices are already defined on your Linux system in the **/etc/dev** directory, starting with **md0**: **/dev/md0** is the first RAID device, and **/dev/md1** is the second, and so on. Each RAID device, in turn, uses hard disk partitions, where each partition contains an entire hard disk. These partitions are usually referred to as RAID disks, whereas a RAID device is an array of the RAID disks it uses.

When creating a RAID partition, you should set the partition type to be **fd**, instead of the 83 for the standard Linux partition. The **fd** type is that used by RAID for automatic detection.

Booting from a RAID Device

As part of the installation process, Fedora lets you create RAID devices from which you can also boot your system. Your Linux system will be configured to load RAID kernel support and automatically detect your RAID devices. The boot loader will be installed on your RAID device, meaning on all the hard disks making up that device.

Fedora does not support booting from RAID 5, only RAID 1. This means that if you want to use RAID 5 and still boot from RAID disks, you will need to create at least two (or more if you want) RAID devices using corresponding partitions for each device across your hard disks. One device would hold your **/boot** partition and be installed as a RAID 1 device. This RAID 1 device would be the first RAID device, **md0**, consisting of the first partition on each hard disk. The second RAID device, **md1**, could then be a RAID 5 device. It would consist of corresponding partitions on the other hard disks. Your system could then boot from the RAID 1 device but use the RAID 5 device.

If you do not create RAID disks during installation, but create them later and want to boot from them, you will have to make sure your system is configured correctly. The RAID devices need to be created with persistent superblocks. Difficulties occur if you are using RAID 5 for your / (root) partition. This partition contains the RAID 5 module, but to access the partition, you have to already load the RAID 5 module. To work around this limitation, you can create a RAM disk in the **/boot** partition that contains the RAID 5 module. Use the **mkinitrd** command to create the RAM disk and the **--with** option to specify the module to include.

```
mkinitrd --preload raid5 --with=raid5 raid-ramdisk 2.6.9-1
```

RAID Administration: mdadm

Red Hat Enterprise Linux and Fedora use the **mdadm** tool to manage and monitor RAID devices. It replaces the older raidtools used on previous versions of Red Hat Enterprise Linux and Fedora. The **mdadm** is an all-purpose tool for creating, monitoring, administering, and fixing RAID devices. You can run commands directly to create and format RAID disks. It also runs as a daemon to monitor and detect problems with the devices.

The **mdadm** tool has seven different modes of operation, each with its own set of options, like monitor with the **-f** option to run it as a daemon, or create with the **-l** option to set a RAID level for a disk. Table 16-2 lists the different modes of operation. Check the **mdadm** Man page for a detailed listing of the options for each mode.

Mode	Description
assemble	Assemble RAID array from devices.
build	Build array without per-device superblocks.
create	Build array with per-device superblocks.
manage	Manage array devices, adding or removing disks.
misc	Specific operations on a device, such as making it read only.
monitor	Monitor arrays for changes and act on them (used for RAID 1, 4, 5, 6).
grow	Change array size, as when replacing smaller devices with larger ones.

Table 16-2: mdadm Modes

Creating and Installing RAID Devices

If you created your RAID devices and their partitions during the installation process, you should already have working RAID devices. Your RAID devices will be configured in the **/etc/mdadm.conf** file, and the status of your RAID devices will be listed in the **/proc/mdstat** file. You can manually start or stop your RAID devices with the **raidstart** and **mdadm** commands. The **-a** option operates on all of them, though you can specify particular devices if you want.

To create a new RAID device manually for an already-installed system, follow these steps:

- Make sure that your kernel supports the RAID level you want for the device you are creating.
- If you have not already done so, create the RAID disks (partitions) you will use for your RAID device.

- Create your RAID device with **mdadm** command in the **build** or **create** mode. The array will also be activated.
- Alternatively, you can configure your RAID device **(/dev/mdn**) in the **/etc/mdadm.conf** file, specifying the RAID disks to use, and then use the mdadm command specifying the RAID device to create.
- Create a file system on the RAID device (**mkfs**) and then mount it.

Adding a separate RAID file system

If you just want to add a RAID file system to a system that already has a standard boot partition, you can dispense with the first RAID 1 partition. Given two hard disks to use for the RAID file system, you would just need two partitions, one for each disk, **/dev/sdb1**, **/dev/sdc1** and **/dev/sdd1**.

```
ARRAY /dev/md0   devices=/dev/sdb1,/dev/sdc1,/dev/sdd1  level=5 num-devices=3
```

You would then create the array with the following command.

```
mdadm -C /dev/md0 -n2 /dev/sdb1 /dev/sdc1 /dev/sdd1  -l5
```

You can then format and mount your RAID device.

Creating Hard Disk Partitions: fd

To add new RAID devices or to create them in the first place, you need to manually create the hard disk partitions they will use, and then configure RAID devices to use those partitions. To create a hard disk partition for use in a RAID array, use **fdisk** or **parted** and specify **fd** as the file system type. You invoke **fdisk** or **parted** with the device name of the hard disk you want to create the partition on. Be sure to specify **fd** as the partition type. The following example invokes **fdisk** for the hard disk **/dev/sdc** (the first hard disk on the secondary IDE connection):

```
fdisk /dev/sdc
```

Though technically partitions, these hard disk devices are referred to as disks in RAID configuration documentation and files.

Note: You can also use gparted or qtparted to create your hard disk partitions. These tools provide and GUI interface for parted (Applications | System Tools menu).

Configuring RAID: /etc/mdadm.conf

Once you have your disks, you then need to configure them as RAID devices. RAID devices are configured in the **/etc/mdadm.conf** file, with options as shown in Table 16-3. This file will be used by the **mdadm** command in the create mode to create the RAID device. In the **/etc/mdadm.conf** file, you create both DEVICE and ARRAY entries. The DEVICE entries list the RAID devices. The ARRAY entries list the RAID arrays and their options. This example implements a simple array on two disks.

```
DEVICE  /dev/sdb1 /dev/sdc1 /dev/sdd1
```

You can list more that one device for a DEVICE entry, as well as have separate DEVICE entries. You can also specify multiple devices using file matching symbols, like *, ?, or []. The following would specify all the partitions on **sde** drive as RAID devices:

```
DEVICE  /dev/sde*  /dev/sdf1
```

For a **ARRAY** entry, you specify the name of the RAID device you are configuring, such as **/dev/md0** for the first RAID device. You then add configuration options such as **devices** to list the partitions that make up the array, **level** for the RAID level, and **num-devices** for the number of devices. The first array holds only the boot partition and uses RAID level 1, whereas the second array uses two partitions and is set to RAID level 5.

```
ARRAY /dev/md0   devices=/dev/sdb1,/dev/sdc1,/dev/sdd1  level=5 num-devices=3
```

The preceding example configures the RAID array **/dev/md0** as three disks (partitions) using **/dev/sdb1, /dev/sdc1,** and **/dev/sdd1**, and is configured for RAID 5 (**level=5**).

Creating a RAID Array

You can create a RAID array either using options specified with the **mdadm** command or using configurations listed in the **/etc/mdadm.conf** file. Use of the **/etc/mdadm.conf** file is not required, though it does make RAID creation more manageable, especially for large or complex arrays. Once you have created your RAID devices, your RAID device will be automatically activated. The following command creates a RAID array, **/dev/md1**, using two devices, **/dev/sdb1**, **/dev/sdc1**, and **/dev/sdd1**, at level 5.

```
mdadm --create /dev/md1 --raid-devices=3 /dev/sdb1 /dev/sdc1 /dev/sdd1 --level=5
```

Directive or Option	Description
DEVICE *devices-list*	Partitions and drives used for RAID devices.
ARRAY	ARRAY configuration section for a particular RAID device.
level=*num*	The RAID level for the RAID device, such as 0, 1, 4, 5, and –1 (linear).
devices=*disk-device-list*	The disk devices (partitions) that make up the RAID array.
num-devices=*count*	Number of RAID devices in an array. Each RAID device section must have this directive. The maximum is 12.
spare-group=*name*	Text name for a spare group, whose devices can be used for other arrays.
auto=*option*	Automatically create specified devices if they do not exist. You can create traditional non-partitioned (**yes** or **md** option) or the newer partitionable arrays (**mdp** or **part** option). For partitionable arrays the default is 4, which you can change.
super-minor	Minor number of the array superblock, same as md device number.
uuid=*UUID-number*	UUID identifier stored in array superblock, used to identify the RAID array. Can be used to reference an array in commands.
MAILADDR	Monitor mode, mail address where alerts are sent.
PROGRAM	Monitor mode, program to run when events occur.

Table 16-3: mdadm.conf Options

Each option has a corresponding short version, as shown in Table 16-4. The same command is shown here with single-letter options.

```
mdadm -C /dev/md1 -n3 /dev/sdb1 /dev/sdc1 /dev/sdd1 -l5
```

If you have configured your RAID devices in the **/etc/mdadm.conf** file, you then use the **mdadm** command in the create mode to create your RAID devices. **mdadm** takes as its argument the name of the RAID device, such as **/dev/md0** for the first RAID device. It then locates the entry for that device in the **/etc/mdadm.conf** file and uses that configuration information to create the RAID file system on that device. You can specify an alternative configuration file with the **-c** option, if you wish. **mdadm** operates as a kind of **mkfs** command for RAID devices, initializing the partitions and creating the RAID file systems. Any data on the partitions making up the RAID array will be erased.

```
mdadm  --create /dev/md0
```

mdadm –create Option	Description
-n --raid-devices	Number of RAID devices
-l --level	RAID level
-C --create	Create mode
-c --chunk	Specify chunk (stripe) size in powers of 2, default is 64KB
-x --spare-devices	Number of spare devices in the array
-z --size	Size of blocks used in devices, by default set to the smallest device if not the same size
-p --parity	Specify the parity algorithm; left-symmetric is used by default

Table 16-4: The mdadm --create Options

Creating Spare Groups

Linux Software RAID now allows RAID arrays to share their spare devices. This means that if arrays belong to the same spare group, then, should a device in one array fail, it can automatically use the spare in another array. Spare devices from any array can be used in another as needed. You set the spare group that an array belongs to with the **--spare-group** option. The **mdadm** monitoring mode will detect a failed device in an array and automatically replace it with a spare device from arrays in the same spare group. The first command in the next example creates a spare drive called **/dev/sde1** for the **/dev/md0** array and labels it **mygroup**. In the second command, array **/dev/md1** has no spare drive but belongs to the same spare group as array **/dev/md0**. Should a drive in **/dev/md1** fail, it can automatically use the spare device, **/dev/sde1**, from **/dev/md0**. The following code lines are really one line, each beginning with **mdadm**.

```
mdadm --create /dev/md0 --raid-devices=4 /dev/sda1 /dev/sdc1 /dev/sdd1 -x
    /dev/sde1 --level=5 --spare-group=mygroup
mdadm --create /dev/md1 --raid-devices=2 /dev/sdf1 /dev/sdg1 --level=1
    --spare-group=mygroup
```

Creating a File System

Once the RAID devices are activated, you can then create file systems on the RAID devices and mount those file systems. The following example creates a standard Linux file system on the **/dev/md0** device:

```
mkfs.ext3 /dev/md0
```

In the following example, the user then creates a directory called **/myraid** and mounts the RAID device there:

```
mkdir /myraid
mount /dev/md0 /myraid
```

If you plan to use your RAID device for maintaining your user directories and files, you would mount the RAID device as your **/home** partition. Such a mounting point might normally be used if you created your RAID devices when installing your system. To transfer your current home directories to a RAID device, first back them up on another partition, and then mount your RAID device, copying your home directories to it.

Managing RAID Arrays

You can manage RAID arrays with the **mdadm** manage mode operations. In this mode you can add or remove devices in arrays, as well as mark ones as failed. The **--add** option lets you add a device to an active array, essentially a hot swap operation.

```
mdadm /dev/md0 --add /dev/sde1
```

To remove a device from an active array, you first have to mark it as failed with the **--fail** option and then remove it with **--remove**.

```
mdadm /dev/md0 --fail /dev/sdb1 --remove /dev/sdb1
```

Starting and Stopping RAID Arrays

To start an already existing RAID array, you use **mdadm** with the assemble mode (newly created arrays are automatically started). To do so directly on the command line requires that you also know what devices make up the array, listing them after the RAID array.

```
mdadm -A /dev/md1 /dev/sdf1  /dev/sdg1
```

It is easier to configure your RAID arrays in the **/etc/mdadm.conf** file. With the scan option, **-s**, **mdadm** will then read array information from the **/etc/mdadm.conf** file. If you do not specify a RAID array, all arrays will be started.

```
mdadm -s /dev/md0
```

To stop a RAID array, you use the **-S** option.

```
mdadm -S /dev/md0
```

Monitoring RAID Arrays

As a daemon, **mdadm** is started and stopped using the **mdmonitor** service script in **/etc/init.d**. This will invoke **mdadm** in the monitor mode, detecting any problems that arise and logging reports as well as taking appropriate action.

```
service mdadm start
```

You can monitor devices directly by invoking **mdadm** with the monitor mode.

```
mdadm --monitor /dev/md0
```

Monitor-related options can be set in the **/etc/mdadm.conf** file. MAILADDR sets the mail address where notification of RAID events are sent. PROGRAM sets the program to use if events occur.

If you decide to change your RAID configuration or add new devices, you first have to deactivate your currently active RAID devices. To deactivate a RAID device, you use the **mdadm** command in the **misc** mode. Be sure to close any open files and unmount any file systems on the device first.

```
umount /dev/md0
mdadm -S /dev/md0
```

fedora

17. Devices

All devices, such as printers, terminals, and CD-ROMs, are connected to your Linux operating system through special files called *device files*. Such a file contains all the information your operating system needs to control the specified device. This design introduces great flexibility. The operating system is independent of the specific details for managing a particular device; the specifics are all handled by the device file. The operating system simply informs the device what task it is to perform, and the device file tells it how. If you change devices, you have to change only the device file, not the whole system.

To install a device on your Linux system, you need a device file for it, software configuration such as that provided by a configuration tool, and kernel support—usually supplied by a module or support that is already compiled and built into the kernel. Device files are not handled in a static way. They are dynamically generated as needed by udev and managed by HAL (Hardware Abstraction Layer). In early versions of Linux and Unix, a device file was created for each possible device, leading to a very large number of device files in the **/etc/dev** directory. Now, your system will detect only those devices it uses and create device files for those only, giving you a much smaller listing of device files. Both udev and HAL are hotplug systems, with udev used for creating devices and HAL designed for providing information about them, as well as managing the configuration for removable devices such as those with file systems such as those for USB card readers and CD-ROMs.

Resource	Description
/sys	The sysfs file system listing configuration information for all the devices on your system
/proc	An older process file system listing kernel information, including device information
www.kernel.org/pub/linux/docs/device-list/devices.txt	Linux device names
www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html	The udev Web site
/etc/udev	The udev configuration directory
http://www.freedesktop.org/wiki/Software/hal	The HAL Web site
/etc/hal	The HAL configuration directory
/usr/share/hal/fdi	The HAL device information files, for configuring HAL information support and policies
/etc/hal/fdi	The HAL system administrator's device information files.

Table 17-1: Device Resources

Managing devices is at the same time easier but much more complex. You now have to use udev and HAL to configure devices, though much of this is now automatic. Device information is maintained in a special device file system called sysfs located at **/sys**. This is a virtual file system like **/proc** and is used to keep track of all devices supported by the kernel. Several of the resources you may need to consult and directories you may have to use are listed in Table 17-1.

The sysfs File System: /sys

The system file system is designed to hold detailed information about system devices. This information can be used by hotplug tools like udev to create device interfaces as they are needed. Instead of having a static and complete manual configuration for a device, the sysfs system is used to maintain configuration information about the device, which is then used as needed by the hotplugging system to created device interfaces when a device is attached to the system. More and more devices are now removable, and many are meant to be attached temporarily (cameras, for example). Instead of maintaining separate static and dynamic methods for configuring devices, Linux distributions make all devices structurally hotplugged.

The sysfs file system is a virtual file system that provides the a hierarchical map of your kernel-supported devices such as PCI devices, buses, and block devices, as well as supporting kernel modules. The **classes** subdirectory will list all your supported devices by category, such as net and sound devices. With sysfs your system can easily determine the device file a particular device is associated with. This is very helpful for managing removable devices as well as dynamically managing and configuring devices as HAL and udev do. The sysfs file system is used by udev to dynamically generate needed device files in the **/dev** directory, as well as by HAL to manage removable device files as needed. The **/sys** file system type is sysfs. The **/sys** subdirectories organized your devices into different categories. The file system is used by **systool** to display a listing of your installed devices. The tool is part of the **sysfsutils** package. The following example will list all your system devices.

```
systool
```

Like **/proc**, the **/sys** directory resides only in memory, but it is still mounted in the **/etc/fstab** file.

The proc File System: /proc

The **/proc** file system is an older file system that was used to maintain information about kernel processes, including devices. It maintains special information files for your devices, though many of these are now supported by the sysfs file system. The **/proc/devices** file lists your installed character and block devices along with their major numbers. IRQs, DMAs, and I/O ports currently used for devices are listed in the **interrupts**, **dma**, and **ioports** files, respectively.

File	Description
/proc/devices	Lists the device drivers configured for the currently running kernel
/proc/dma	Displays the DMA channels currently used
/proc/interrupts	Displays the IRQs (interrupts) in use
/proc/ioports	Shows the I/O ports in use
/proc/asound	Lists sound devices
/proc/net	Directory for network devices

Table 17-2: Proc Device Information Files

Certain files list information covering several devices, such as **pci**, which lists all your PCI devices, and **sound**, which lists all your sound devices. The **sound** file lists detailed information

about your sound card. Several subdirectories, such as **net** and **scsi**, contain information files for different devices. Certain files hold configuration information that can be changed dynamically, such as the IP packet forwarding capability and the maximum number of files. You can change these values with the **sysctl** tool or by manually editing certain files. Table 17-2 lists several device-related **/proc** files.

udev: Device Files

Devices are now treated as *hotpluggedable,* meaning they can be easily attached and removed. Their configuration is dynamically detected and does not rely on manual administrative settings. The hotplug tool used to detect device files is udev, user devices. Each time your system is booted, udev will automatically detect your devices and generate device files for them in the **/etc/dev** directory. This means that the **/etc/dev** directory and its files are recreated each time you boot. It is a dynamic directory, no longer static. udev uses a set of rules to direct how device files are to be generated, including any corresponding symbolic links. These are located in the **/etc/udev/rules.d** and **/lib/udev/rules.d** directories. You can find out more about udev at **http://kernel.org/pub/linux/utils/kernel/hotplug/udev.html**.

As part of the hotplug system, udev will automatically detect kernel devices that are added or removed from the system. When the device interface is first created, its corresponding **sysfs** file is located and read, determining any additional attributes such as serial numbers and device major and minor numbers that can be used to uniquely identify the device. These can be used as keys in udev rules to create the device interface. Once the device is created, it is listed in the udev database, which keeps track of currently installed devices.

If a device is added, udev is called by hotplug. It checks the sysfs file for that device for the major and minor numbers, if provided. It then uses the rules in its rules file to create the device file and any symbolic links to create the device file in **/dev**, with permissions specified for the device in the udev permissions rules.

Note: When the system starts, it invokes **/sbin/udevstart**, which runs udev and creates all the kernel devices making device files in the **/dev** directory.

As **/dev** is now dynamic, any changes you would make manually to the **/dev** directory will be lost when you reboot. This includes the creation of any symbolic links such as **/dev/cdrom** that many software applications use. Instead, such symbolic links have to be configured udev rules files located in the **/etc/udev/rules.d** and **/lib/udev/rules.d** directories. Default rules are already in place for symbolic links, but you can create rules to add your own.

udev Configuration

The configuration file for udev is **/etc/udev/udev.conf**. Here are set global udev options such as the logging level. The udev tool uses the udev **rules.d** files to dynamically create your device files. Be very careful in making any changes, particularly to rules file locations. Support for all devices on your system relies on these rules. The default **udev.conf** file is shown here and supports entries only for the location of the device files (**udev_root**) and the logging priority (**udev_log**).

```
# udev.conf
# The initial syslog(3) priority: "err", "info", "debug" or its
# numerical equivalent. For runtime debugging, the daemons internal
# state can be changed with: "udevadm control --log_priority=<value>".
udev_log="err"
```

You use the **udevadm** command to obtain information about devices and to set options for the **udev** server, primarily options used in debugging. The **udevadm** control command has options to set the log priority, stop processing events, set environment variables, and set the number or maximum events allowed. **udevadm** takes as its argument a command, which in turn may have options. The commands are listed in Table 17-3.

Command	Description
info *options*	Queries udev database for device information
settle *options*	Watch udev device queue
control *options instructions*	Set options for udev events
monitor *options*	Display udev events
test *options devpath*	Simulate udev event for testing and debugging
version	Udev version
help	List udev commands and description
trigger *options*	Request device events

Table 17-3: udevman commands

Device Names and udev Rules: /etc/udev/rules.d and /lib/udev/rules.d

The name of a device file is designed to reflect the task of the device. Printer device files begin with **lp** for "line print." Because you could have more than one printer connected to your system, the particular printer device files are distinguished by two or more numbers or letters following the prefix **lp**, such as **lp0**, **lp1**, **lp2**. The same is true for terminal device files. They begin with the prefix **tty**, for "teletype," and are further distinguished by numbers or letters such as **tty0**, **tty1**, **ttyS0**, and so on. You can obtain a complete listing of the current device filenames and the devices for which they are used from the **www.kernel.org** web site at **www.kernel.org/pub/linux/docs/device-list/devices.txt**.

With udev, device names are determined dynamically by rules listed in the udev rules files. These are located in **/etc/udev/rules.d** and in **/lib/udev/rules.d**. The rules files are named, beginning with a number to establish priority. They are read sequentially, with the first rules overriding any conflicting later ones. All rules files have a **.rules** extension.

The **/etc/udev/rules.d** directory hold specialized rules for your installation that you can configure. The **/lib/udev/rules.d** directory holds system rules, including default rules, which you do not modify. The rules files that you will find in the **/lib/udev/rules.d** directory are generated by your system during installation. You should never edit them. If you need to add rules of your own, you should create your own rules file, or carefully edit the rules files in the **/etc/udev/rules.d** directory.

/etc/udev/rules.d

In the **/etc/udev/rules.d** directory, rules have been distributed among a variety of different rules files which can be modified. Network device rules are held in the **70-persistent-net.rules** file, and the rules for DVD/CD devices are held in the **70-persistent-cd.rules** file. Some rules files are set up for more specialized devices like **60-rules-libmtp** for music players, **60-pcmcia.rules** for PCMCIA devices, and **90-alsa.rules** for the sound driver**.** These rules files can be modified by adding additional rules. You should be careful when performing any modifications, making sure your rules are correct.

To customize your setup, you can create your own separate rules files in **/etc/udev/rules.d**. In your rules file you would normally define only symlinks, using SYMLINK keys alone, as described in the following sections. These set up symbolic links to devices, letting you access them with other device names. NAME keys are used to create the original device interface, a task usually left to udev.

/lib/udev/rules.d

Default rules for your devices are placed by udev in the rules files in the **/lib/udev/rules.d/** directory. These rules have been provided for your Fedora distribution and are designed specifically for it. You should never modify these rules. They will always be overwritten on update.

Though you never edit these files, though you can check them to see how device naming is handled. These files will create device files using the official kernel names. These names are often referenced directly by applications that expect to find devices with these particular names, such as **lp0** for a printer device

udev rules

In an udev rules file, each line maps a device attribute to a device name, as well as specifying any symbolic names (links). Attributes are specified using keys, of which there may be more than one. If all the keys match a device, then the associated name is used for it and a device file of that name will be generated. An assignable key, like NAME for the device name or SYMLINK for a symbolic name, is use to assign the matched value. Instead of listing a device name, a program or script may be specified instead to generate the name. This is often the case for DVD/CD-ROM devices, where the device name could be a dvdwriter, cdwriter, cdrom, or dvdrom.

The rules consist of a comma-separated list of fields. A field consists of a matching or assignable key. The matching keys use the == and **!=** operators to compare for equality and inequality. The *****, **?**, and **[]** operators can match any characters, any single character, or a class of characters, just as in the shell. The assignable keys can use the =, +=, and **:=** operators to assign values. The = operator assigns a single value, the += appends the value to those already assigned, and the **:=** operator makes an assignment final, preventing later changes. The assignable keys also support. The udev keys are listed in Table 17-4. Check the udev Man page for detailed descriptions.

The key fields such as **KERNEL** support pattern matching to specify collections of devices. For example, **mouse*** will match all devices beginning with the pattern “mouse”. The following field uses the **KERNEL** key to match on all mouse devices as listed by the kernel:

```
KERNEL="mouse*"
```

Matching Keys	Description
ACTION	Match the event action
DEVPATH	Match the device path
ENV{key}	Match an environment variable value
BUS	Match the bus type of the device. (The sysfs device bus must be able to be determined by a "device" symlink.)
DRIVER	Match the device driver name.
ID	Match the device number on the bus, for instance, the PCI bus ID.
KERNEL	Match the kernel device name.
PROGRAM	Use an external program to determine the device. This key is valid if the program returns successful. The string returned by the program may be additionally matched with the RESULT key.
RESULT	Match the returned string of the last PROGRAM call. This key may be used in any following rule after a PROGRAM call.
SUBSYSTEM	Match the device subsystem.
SYSFS{{filename}}	Match the sysfs device attribute, for instance, a label, vendor, USB serial number, SCSI UUID, or file system label.
Assignable Keys	**Description**
NAME	The name of the node to be created, or the name the network interface should be renamed to.
OWNER, GROUP, MODE	The permissions for the device.
PLACE	Match the location on the bus, such as the physical port of a USB device.
ENV{key}	Export variable to environment
IMPORT{type}	Import results of a program, contents of a text file, or stored keys in a parent device. The type can be program, file, or parent.
SYMLINK	The name of the symbolic link (symlink) for the device.
RUN	Add program to list of programs to be run by device

Table 17-4: udev Rule Keys

The next key will match on all printer devices numbered **lp0** through **lp9**. It uses brackets to specify a range of numbers or characters, in this case 0 through 9, **[0-9]**:

```
KERNEL="lp[0-9]*"
```

The **NAME**, **SYMLINK**, and **PROGRAM** keys support string substitution codes similar to the way printf codes work. Such a code is preceded by a **%** symbol. The code allows several possible devices and names to be referenced in the same rule. Table 17-5 lists the supported codes.

Substitution Code	Description
%n	The kernel number of the device
%k	The kernel name for the device
%M	The kernel major number
%m	The kernel minor number
%p	The path of the device
%b	Device name matched from the device path
%c	The string returned by a PROGRAM key (can't be used in a PROGRAM key)
%s {*filename*}	Content of sysfs attribute
%%	Quotes the % character in case it is needed in the device name.
%E{*key*}	Value of environment variable
%N	Name of a temporary device node, to provide access before real node is created

Table 17-5: udev Substitution Codes

Rules primarily use **KERNEL** keys to designate devices. The **KERNEL** key is followed by either a **NAME** key to specify the device filename or a **SYMLINK** key to set up a symbolic link for a device file. The following rule uses the **KERNEL** key to match on all mouse devices as listed by the kernel. Corresponding device names are placed in the **/dev/input** directory, and the name used is the kernel name for the device (**%k**):

```
KERNEL="mouse*", NAME="input/%k"
```

A default rule for mice can be found in the **/lib/udev/ruled.d/50-udev-default.rules** file. It covers mouse and mice devices. The MODE is 0640, which is discussed in later sections.

```
KERNEL=="mouse*|mice|event*", NAME="input/%k", MODE="0640"
```

This rule uses both a **SUBSYTEM** key and a **KERNEL** key to set up device files for USB printers, whose kernel names will be used to create device files in **/dev/usb**. SYMLINKs are set up at the same time with the prefix **/dev/usb**.

```
SUBSYSTEM=="usb", KERNEL=="lp*", NAME="usb/%k", SYMLINK+="usb%k", GROUP="lp"
```

Symbolic Links

Certain device files are really symbolic links bearing common device names that are often linked to the actual device file used. A *symbolic link* is another name for a file that is used like a shortcut, referencing that file. Common devices like printer, CD-ROM, hard drives, SCSI devices, and sound devices, along with many others, will have corresponding symbolic links. For example, a **/dev/cdrom** symbolic link links to the actual device file used for your CD-ROM. If your CD-ROM is an IDE device, it may use a device file such as **sdc**. In this case, **/dev/cdrom** would be a link to **/dev/sdc**. In effect, **/dev/cdrom** is another name for **/dev/sdc**. Serial ATA DVD/CD drives will be linked to **scd** devices, like **scd0** for the first Serial ATA CD/DVD drive. If you drive functions both as a CD and DVD writer and reader, you will have several links to the same device. In this

case the links **cdrom**, **cdrw**, **cdwriter**, **dvd**, **dvdrw**, **dvdwriter** will all link to the same CD/DVD RW-ROM device.

A **/dev/modem** link file also exists for your modem. If your modem is connected to the second serial port, its device file would be **/dev/ttyS1**. In this case, **/dev/modem** would be a link to that device file. Applications can then use **/dev/modem** to access your modem, instead of having to know the actual device file used. Table 17-6 lists commonly used device links.

Symbolic links are created by udev using the SYMLINK key. The symbolic links for a device can be listed either with the same rule creating a device file (NAME key) or in a separate rule that will specify only a symbolic link. The inclusion of the NAME key does not have to be specific, if the default device name is used. The + added to the = symbol will automatically create the device with the default name, not requiring an explicit NAME key in the rule. The following rule is for parallel printers. It includes both the default name, and implied NAME key creating the device (+), and a symbolic link, **par**. The **%n** will add a number to the symbolic link like **par1**, **par2**, and so on. The rule can be found in the **/lib/udev/rules.d/50-udev-default.rules** file.

```
KERNEL=="lp[0-9]*",  GROUP="lp", SYMLINK+="par%n"
```

Link	Description
cdrom	Link to your CD-ROM device file
dvd	Link to your DVD-ROM device file
cdwriter	Link to your CD-R or CD-RW device file,
dvdwriter	Link to your DVD-R or DVD-RW device file
modem	Link to your modem device file
floppy	Link to your floppy device file
tape	Link to your tape device file
scanner	Link to your scanner device file
mouse	Link to your mouse device file
tape	Link to your tape device file

Table 17-6: Device Symbolic Links

If you want to create more than one symbolic link for a device, you can list them in the SYMLINK key. Should you decide to set up a separate rule that specifies just a symbolic link, the symbolic link will be kept on a list awaiting the creation of its device. This also allows you to add other symbolic links for a device in other rules files. This situation can be confusing because symbolic links can be created for devices that are not yet generated. The symbolic links will be defined and held until needed, when the device is generated. This is why you could have many more SYMLINK rules than NAME rules in udev that actually set up device files. In the case of removable devices, they will not have a device name generated until they are connected.

In most cases you will only need symbolic links for devices using the official symbolic names. Most of these are already defined for you. Should you need to create just a symbolic link, you can create a SYMLINK rule for it. However, a new SYMLINK rule needs to be placed before the name rules that name that device. The SYMLINK rules for a device are read by udev and kept until a device is named. Then those symbolic names can be used for that device. You can have as

many symbolic links for the same device as you want, meaning that you could have several SYMLINK rules for the same device. When the NAME rule for the device is encountered, the previous SYMLINK keys are simply appended.

Most standard symbolic names are already defined in the rules files, such as **video** for the video device. In the following example, the device is referenced by its KERNEL key and the symbolic link is applied with SYMLINK key. This is only a SYMLINK rule. The NAME key is implied:

```
KERNEL="video0", SYMLINK+="video"
```

Symbolic links for DVD/CD and Network devices are generated as part of their rules in their respective rules files.

Program Keys, IMPORT{program} keys, and /lib/udev

The PROGRAM, RUN, and IMPORT keys can be used to specify and run external scripts or programs needed to set up or manage devices. The PROGRAM keys is used for scripts that set up devices, and the RUN key to run programs on devices already set up. RUN programs are usually executed when a device is removed or attached. The IMPORT key is designed to import results or file content into a rules file. It can also be used to run programs to generate rules for devices. The IMPORT and PROGRAM keys normally use udev callouts that return device values like serial numbers or perform rule generation tasks (see Table 17-7).

Note: A program can be invoked either with the **PROGRAM** key or the **IMPORT**[*program*} key. The **PROGRAM** key will run the program and return 0 if successful. Returned values can be held in the **RESULTS** key. The **IMPORT{program}** key returns assignable values

The PROGRAM scripts and programs used to set up devices can be found in the **/lib/udev** directory. These include **write_cd_rules** for setting up CD/DVD-RW/ROM symlinks, **create_floppy_devices** for floppy drives, and various ID scripts for determining the serial number of a device like **usb_id**, **vol_id**, and **cdrom_id**.

The rules in the **75-cd-aliases-generator.rules** file in the **/lib/udev/rules.d** directory are used to generate symlinks for CD/DVD-ROMs that are not covered by any previous rules. They are designed for CD/DVD devices that are frequently being removed and attached at different connections. These rules, including the following one, will invoke the **write_cd_rules** script in the **/lib/udev** directory, using the PROGRAM key.

```
ACTION=="add", SUBSYSTEM=="block", ENV{ID_CDROM}=="?*", ENV{GENERATED}!="?*",
PROGRAM="write_cd_rules", SYMLINK+="%c"
```

The **write_cd_rules** script will generate a rule in the appropriate rules file in the **/etc/udev/rules.d** directory. These rules files are editable (unlike those in the **/lib/udev/rules.d** directory).

The IMPORT command will import the results of a program or the content of file into the rule file. The IMPORT options are **program** for a program to run, or **file** for the contents of a file to read. In some cases it used primarily to run a script to generate rules. IMPORT operation are used to determine the serial number of a removable device using scripts like **ata_id**. Attached devices will have serial numbers accessible in the **/dev/disk/by-id** directory.

```
KERNEL=="hd*[!0-9]", IMPORT{program}="ata_id --export $tempnode"
```

An IMPORT operation is used for persistent network devices, the rules in the **75-persistent-net-gnerator.rules** file uses the **write_net_rules** script to generate rules. This script both write rules to the **/etc/dev/ruled.d/70-peristent-net.rules** file.

```
# write rule
DRIVERS=="?*", IMPORT{program}="write_net_rules"
```

For hotplug network connections, the **60-net.rules** file will run a rule to run the **net.hotplug** script using the RUN key.

```
SUBSYSTEM=="net", RUN+="/etc/sysconfig/network-scripts/net.hotplug"
```

udev callout	Description
ata_id	Callout script to return a serial number for a ATA storage device
cdrom_id	Callout script to return a serial number for a CD/DVD-ROM device
scsi_id	Callout script to return a serial number for a SCSI storage device
path_id	Callout script to generate a unique path name for a device
ata_id	Callout script to return a serial number for a ATA storage device
usb_id	Callout script to return a serial number for a USB device
vol_id	Callout script to return a serial number for a Volume storage device
create_floppy_devices	Create a floppy device
rename_device	Rename a device
write_cd_rules	Write persistent CD/DVD-RW/ROM rules
write_net_rules	Write persistent network device rules

Table 17-7: Udev callouts (scripts), /lib/udev

Creating udev Rules

If you want to create rules of your own, you can place them in a separate rules file in the **/etc/udev/rules.d** directory (though you can edit the rules files already in that directory). The NAME rules that name devices are read lexically, where the first NAME rule will take precedence over any later ones. Only the first NAME rule for a device will be used. Later NAME rules for that same device will be ignored. Keep in mind that a SYMLINK rule with a += includes a NAME rule for the default device, even though the NAME key is not explicitly shown. Check www.reactivated.net/writing_udev_rules.html for a tutorial on writing udev rules.

Since rules are being created that are meant to replace the default rules, they would have to be run first. To do this, you would place them in a rules file that begins with a very low number, say 10. Rules files are read in lexical order, with the lower numbers read first. You could create a file called **10-user.rules** in the **/etc/udev/rules.d** directory. Here you would place your own rules. Conversely, if you wanted rules that would run only if the defaults failed for some reason, you would use a rules file numbered higher, like **90-mydefaults.rules**. For example, a user could set up a canon-pr rule to replace the default printer rule for that printer. The new user canon-pr rule would

be placed in a **10-user.rules** file to be executed before the printer rules file, thereby taking precedence. The default printer rule (shown here) would not be applied to the Canon printer.

```
SUBSYTEM="usb", KERNEL="lp*", NAME="usb/%k"
```

SYMLINK Rules

In most cases, you will only need to create symbolic links for devices, using the official name. You can also create rules that just create symbolic links. However, these need to be placed before the name rules that name the devices. These SYMLINK rules are read by udev and kept until a device is named. Then all the symbolic names will be used for that device. You can have as many symbolic links for the same device as you want, meaning that you could have several SYMLINK rules for the same device. When the NAME rule for the device is encountered, the previous SYMLINK keys are simply appended.

Most standard symbolic names are already defined, such as audio for the audio device. In the following example, the device is referenced by its KERNEL key and the symbolic link is applied with SYMLINK key. The NAME key for the default device is implied:

```
KERNEL="video0",  SYMLINK+="video"
```

If you always know the name for a device, you can easily add a SYMLINK rule. For example, if you know your DVD-ROM is attached to the first secondary IDE connection (**sdc**), you can create a symbolic name of your own choosing with a SYMLINK rule. In the next example a new symbolic link, **mydvdrom**, is created for the DVD-ROM on the **/dev/sdc** device.

```
KERNEL="sdc", SYMLINK="mydvdrom"
```

For a SYMLINK rule to be used, it must occur before a NAME rule that names the device. You should place these rules in a file that will precede the other rules files, such as **10-user.rules**.

Persistent Names: udevinfo

The default udev rules will provide names for your devices using the official symbolic names reserved for them, for instance, lp*n* for printer, where *n* is the number of the printer. For fixed devices, such as fixed printers, this is normally adequate. However, for removable devices, such as USB printers, that may be attached in different sequences at different times to USB ports, the names used may not refer to the same printer. For example, if you have two USB printers, an Epson and Canon, and attach the Epson first and the Canon second, the Epson will be given the name **usb/lp0** and the Canon will have the name **usb/lp1**. If, however, you later detach them and reattach the Canon first and the Epson second, then the Canon will have the name **usb/lp0** and the Epson will have **usb/lp1**. If you want the Epson to always have the same symbolic name, say **epson-pr**, and likewise the Canon, as in **canon-pr**, you would have to create your own rule for detecting these printers and giving them your own symbolic names.

The key task is creating a persistent name is to use unique information to identify the device. You then create a rule that references the device with the unique information, identifying it, and then name it with an official name, but giving it a unique symbolic name. You can then use the unique symbolic name, like **canon-pr**, to always reference just that printer and no other, when it is plugged in. In this example, unique information such as the Canon printer serial number is used to identify the Canon printer. It is next named with the official name, **usb/lp0** or **usb/lp1**, depending on whether another printer was plugged in first, and then it is given a unique symbolic name,

canon-pr, which will reference that official name, whatever it may be. Keeping the official name, like **lp0**, preserves standard access to the device as used by many applications.

You use **/sys** file system information about the device to detect the correct device to reference with the symbolic link. Unique **/sys** device information such as the vendor serial number or the vendor name can be used to uniquely reference the device. To obtain this information, you need to first query the **/sys** file system. You do this with the **udevinfo** command.

First you will need to know where the device is located in the **/sys** file system. You plug in your device, which will automatically configure and name it, using the official name. For example, plugging in the USB printer will create a **/dev/usb/lp0** device name for it. You can use this device name to find out where the USB printer information is in **/sys**. Use the **udevinfo** command with the **-q path** option to query for the **/sys** pathname, and add the **-n** option with the device's full pathname to identify the device you are searching for. The following command will display the **/sys** path for the printer with the device name **lp0**. In this case, the device is in the **class** subdirectory under **usb**. The path will assume **/sys**.

```
udevinfo -q path -n  /dev/usb/lp0
  /class/usb/lp0
```

Once you have the device's **/sys** path, you can use that path to display information about it. Use the **udevinfo** command again with the **-a** option to display all information about the device and the **-p** option to specify its path in the **/sys** file system. The listing can be extensive, so you should pipe the output to **less** or redirect it to a file.

```
udevinfo -a  -p  /sys/class/usb/lp0 | less
```

Some of the key output to look for is the BUS used and information such as the serial number, product name, or manufacturer. Look for information that would uniquely identify the device, such as serial number. Some devices will support different buses, and the information may be different for each. Be sure to use the information for that bus when setting up your keys in the udev rule.

```
BUS="usb"
ATTRS{serial}="300HCR"
ATTRS{manufacturer}="Canon"
ATTRS{idproduct}="1074"
ATTRS{product}="S330"
```

You can use much of this information in a ATTRS key in an udev rule to identify the device. The ATTRS key (attributes) is used to obtain **/sys** information about a device. You use the ATTRS key with the field you want referenced placed in braces. You can then match that field to a value to reference the particular device you want. Use the = sign and a valid field value to match against. Once you know the **/sys** serial number of a device, you can use it in ATTRS keys in udev rules to uniquely reference the device. The following key checks the serial number of the devices field for the Canon printer's serial number:

```
ATTRS{serial}="300HCR"
```

A user rule can now be created for the Canon printer.

In another rules file you can add your own symbolic link using **/sys** information to uniquely identify the printer, and name the device with its official kernel name. The first two keys, SUBSYSTEM and ATTRS, specify the particular printer. In this case the serial number of the

printer is used to uniquely identify it. The NAME key will name the printer using the official kernel name, always referenced with the **%k** code. Since this is a USB printer, its device file would be placed in the **usb** subdirectory, **usb/%k**. Then the SYMLINK key defines the unique symbolic name to use, in this case **canon-pr** in the **/dev/usb** directory.

```
SUBSYSTEM="usb", ATTRS{serial}="300HCR", NAME="usb/%k", SYMLINK="usb/canon-pr"
```

The rules are applied dynamically in real time. To run a new rule, simple attach your USB printer (or detach and reattach). You will see the device files automatically generated.

Permission and Owner Fields: MODE, GROUP, OWNER

Additional fields provide ownership and permission attributes for devices. The MODE field is used to specify permissions, the OWNER to specify the user the owns the device (usually the root user). The GROUP field lets you add the device to a group that can then be used to control access.

The MODE field is a octal bit permission setting, the same as used for file permissions. Usually this is set to 660, owner and group read/write permission. Pattern matching is supported with the *****, **?**, and **[]** operators. The following example sets mouse devices to the owner read/write owner and group read permissions, 0640:

```
KERNEL=="mouse*|mice|event*", NAME="input/%k", MODE="0640"
```

The floppy device (**fd**) entry specifies a **floppy** group.

```
KERNEL=="fd[01]*", GROUP="floppy"
```

Parallel printer devices (**parport**) are assigned to the **lp** group.

```
KERNEL=="parport[0-9]*", GROUP="lp"
```

Tape devices (**pt** and **ntp**) use the **disk** group.

```
KERNEL==""pt[0-9]*|npt[0-9]*", GROUP="disk"
```

The default settings set the OWNER and GROUP to root with owner read/write permissions (600).

```
KERNEL=="*",  OWNER="root" GROUP="root", MODE="0600"
```

Hardware Abstraction Layer: HAL

The purpose of the Hardware Abstraction Layer (HAL) is to abstract the process of applications accessing devices. Applications should not have to know anything about a device, even its symbolic name. It should just have to request a device of a certain type, and then a service, like HAL, should provide what is available. With HAL, device implementation becomes hidden from applications.

HAL makes devices easily available to desktops and applications using a D-BUS (device bus) structure. Device are managed as objects that applications can easily access. The D-BUS service is provided by the HAL daemon, **haldaemon**. Interaction with the device object is provided by the **www.freedesktop.org** HAL service.

HAL is an information service for devices. The HAL daemon maintains a dynamic database of connected hardware devices. This information can be used by specialized callout

programs to maintain certain device configuration files. This is the case with the managing removable storage devices. HAL will invoke the specialized callout programs that will use HAL information to dynamically manage devices. Removable devices like CD-ROM discs or USB card readers are managed by specialized callouts with HAL information, detecting when such disks are attached. The situation can be confusing. Callout programs perform the actual tasks, but HAL provides the device information. For example, though the callout hal-system-storage-mount mounts a device, the options and mountpoints used for CD-ROM entries are specified in HAL device information files that set policies for storage management.

Note: No longer are entries maintained in the **/etc/fstab** file for removable devices like your CD-ROMs. These devices are managed directly by HAL using it set of storage callouts like **hal-storage-mount to mount** a device or **hal-storage-eject** to remove one. In effect you now have to use the HAL device information files to manage your removable file systems.

HAL is a software project of **www.freedesktop.org**, which specializes in open source desktop tools. Check the latest HAL specification documentation at **http://freedesktop.org** (search on **HAL**) for detailed explanations of how HAL works (the **specifications** link on the HAL page, Latest HAL Specification). The documentation is very detailed and complete. Also download and install the **hal-docs** package. In the **/usr/share/doc/hal*** directory you will find detailed documentation for HAL specifications (**spec/hal-spec.html**), as well and a general overview. Use your browser to read the **hal-spec.html** page (the version number, **0.5.12**, will change with HAL updates).

```
file:///usr/share/doc/hal-0.5.12/spec/hal-spec.html
```

The HAL Daemon and HAL tools

The HAL daemon, **hald**, is run as the **haldaemon** process. Information provided by the HAL daemon for all your devices can be displayed using the HAL tools such as **lshal** to list your HAL devices and **hal-get-property** to display a certain property of a device. The **hal-get-property** tool used the **--udi** option to obtain a device's Unique Device Identifier and the **--key** option for the name of the property.

The **hal-set-property** usee the **--udi** and **--key** options. You can set property values with options like **--string**, **--bool**, **--int**, and add values with options like **--strlist-post**. use the **--remove** option to remove a property.

The **hal-find-by-property** lets you search for a HAL device by searching device properties. The **--string** option lets you provide a pattern to search for in the HAL device property fields. The **--key** option will search on a name. The UIDs for matching devices are output. The hal-find-by-capability lets you search for a HAL device based on device capabilities. The **--capability** option lets you provide a pattern to search for. The UIDs for matching devices are displayed.

For removable media, HAL will continually poll for the device in order to quickly detect it when inserted or attached. This can cause problems if the device configuration is corrupted, as well as apply addition overhead for managing the device. Should you have difficulties with a removable media device, you can disable polling for it with hal-disable-polling. The **--udi** option uses the UDI to reference the device. You can also use the **--enable-polling** option to restore polling.

The **lock-out** tool let you control HAL locks manually. It is useful only for software that does not use HAL already. The **--run** option specifies the program to be run.

Tool	Description
`lshal`	Display HAL devices
`hal-get-property`	Retrieve a device property for the HAL database
`hal-set-property`	Set a device property for the HAL database
`hal-find-by-property`	Search for a device on the property fields
`hal-find-by-capability`	Search for a device based on a capability
`hal-disable-polling`	Disable polling for removable media.

HAL Configuration: /etc/hal/fdi, and /usr/share/hal/fdi

Information about devices and policies to manage devices are held in device information files in the **/etc/hal/fdi** and **/usr/share/hal/fdi** directories. These directories are where you would set properties such as options that are to be used for CD-ROMs in **/etc/fstab**.

The implementation of HAL on Linux configures storage management by focusing on storage methods for mountable volumes, instead of particular devices. Volume properties define actions to take and valid options that can be used. Special callouts perform the actions directly, such as **hal-storage-mount** to mount media, or **hal-storage-eject** to remove it.

Device Information Files: fdi

HAL properties for these devices are handled by device information files (fdi) in the **/usr/share/hal/fdi** and **/etc/hal/fdi** directories. The **/usr/share/hal/fdi** directory is used for configurations provided by the distribution, whereas **/etc/hal/fdi** is used for setting user administrative configurations. In both are listed subdirectories for the different kinds of information that HAL manages, such as **policy**, whose subdirectories have files with policies for how to manage devices. The files, known as device information files, have rules for obtaining information about devices, as well as detecting and assigning options for removable devices. The device information files have the extension **.fdi**, as in **storage-methods.fdi**. For example, the **policy** directory has two subdirectories: **10osvendor** and **20thirdpary**. The **10osvendor** holds the fdi files that have policy rules for managing removable devices on Linux (10osvendor replaces 90defaultpolicy in earlier HAL versions). This directory holds the **20-storage-methods.fdi** policy file used for storage devices. Here you will find the properties that specify options for removable storage devices such as CD-ROMs. The directories begin with numbers; lower numbers are read first. Unlike with udev, the last property read will override any previous property settings, so priority is given to higher-numbered directories and the fdi files they hold. This is why the default policies are in **10osvendor**, whereas the user policies, which would override the defaults, would be in a higher-numbered directory like **30user**, as are third-party policies, **20thirdpolicy**.

There are currently three device information file directories set up in the device information file directories, each for different kinds of information: information, policy, and preprobe.

- **Information** The **information** directory is for information about devices.
- **Policy** The **policy** directory is for setting policies such as storage policies. The default policies for a storage device are in a **20-storage-methods.fdi** file in the **policy/10osvendor** directory.
- **Preprobe** The **preprobe** directory handles difficult devices such as unusual drives or drive configurations, for instance, those in **preprobe/10osvendor/10-ide-drives.fdi**. This contains information needed even before the device is probed.

Within these subdirectories are still other subdirectories indicating where the device information files come from, such as **vendor**, **thirdparty**, or **user**, and their priority. Certain critical files are listed here:

- **information/10freedesktop** Information provided by freedesktop.org
- **policy/10osvendor** Default policies (set by system administrator and OS distribution)
- **preprobe/10usevendor** Preprobe policies for difficult devices

Properties

Information for a device is specified with a *property* entry. Such entries consist of a key/value pair, where the key specifies the device and its attribute and the value is the value for that attribute. There are many kinds of values, such as Boolean true/false, string values such as those use to specify directory mountpoints, or integer values.

Properties are classified according to metadata, physical connection, function, and policies. Metadata provides general information about a device, such as the bus it uses, its driver, or its HAL ID. Metadata properties begin with the key info, as in info.bus. Physical properties describe physical connections, namely the buses used. The PCI and SCSI bus information is listed in **pci** and **scsi** keys. The **usb_device** properties are used for the USB bus.

The functional properties apply to specific kinds of devices. Here you will find properties for storage devices, such as the **storage.cdrom** keys that specify if an optical device is writable capabilities. For example, the **storage.cdrom.cdr** key set to true will specify that an optical drive can write to CD-R discs.

The volume properties are used for devices that can be mounted, like CD discs or USB drives.

Methods and properties can indicate how devices are to be handled. Methods are, in effect, the directives that callout programs will use to carry out tasks. Methods for storage media are handled using Volume properties, specifying methods (callouts) to use. Volume properties can specify device options like file system type and mount options. HAL uses scripts in the **/usr/libexec** directory to actually manage media. The following abbreviated entries come from the **20-storage-methods.fdi** policy file. The first specifies the action to take and the second the callout script to execute, **hal-storage-mount**.

```
<append key="Volume.method_names" type="strlist">Mount</append>
<append key="Volume.method_execpaths" type="strlist">hal-storage-mount</append>
```

Mount options are designated using volume.mount.valid_options as shown here for ro (read only). Options that will be used will be determined when the mount callout is executed.

```
<append key="volume.mount.valid_options"type="strlist">ro</append>
```

Several of the commonly used volume methods and properties are listed in Table 17-8.

Property	Description
volume.method_names	Action to be taken
volume.method.execpath	Callout script to be run for a device
volume.mount_point (*string*)	The preferred mountpoint for the storage device
volume.mount_filesystem (*string*)	File system to use when mounting a volume
volume.mount.mount_options (*strlist*)	Default mount options for volumes, where ***strlist*** can be any mount option, such as **ro** or **exec**

Table 17-8: HAL Volume properties and methods.

Device Information File Directives

Properties are defined in directives listed in device information files. As noted, device information files have **.fdi** extensions. A directive is encased in greater- and less-than symbols. There are three directives:

- The **merge** directive will merge a new property into a device's information database.
- The **append** directive will append or modify a property for that device already in the database.
- The **match** directive will test device information values.

A directive will include a type attribute designating the type of value to be stored such as string, bool, int, and double. The **copy_property** type will copy a property. The following discussion of the **storage-methods.fdi** file shows several examples of merge and match directives.

storage.fdi

The **20-storage-methods.fdi** file in the **/usr/share/hal/fdi/policy/10osvendor** directory lists the policies for your removable storage devices. Here is where your options for storage volumes (e.g., CD-ROM) entries are actually specified. The file is organized in sections beginning with particular types of devices to standard defaults. Keys are used to define options, such as **volume.mount.valid_options**, which will specify a mount option for a storage device such as a CD-ROM. Keys are used to specify exceptions like hotplugged devices.

The **20-storage-methods.fdi** file begins with default properties and then lists those for specific kinds of devices. Unless redefined in a later key, the default will remain in effect. The options you will see listed for the default storage volumes will apply to CD-ROMs. For example, the **noexec** option is set as a valid default. The following sets **noexec** as a default mount option for a storage device. There are also entries for **ro** and **quiet**. The append operation adds the policy option.

```
<append key="volume.mount.valid_options"type="strlist">noexec</append>
```

The default mountpoint root directory for storage devices is now set by the mount callout script, **hal-storage-mount**. Currently this is **/media**. The default mountpoint is disk. HAL will try to use the Volume property information to generate a mountpoint.

The following example manages blank disks. Instead of being mounted, such disks can only be ejected. To determine possible actions, HAL uses **method_names**, **method_signatures**, and **method_execpaths** for the Volume properties (the org.freedesktop.Hal prefix for the keys has been removed from this example to make it more readable, as in **org.freedesktop.Hal.Volume.method_names**).

```
<match key="volume.disc.is_blank" bool="true">

<append key="info.interfaces"type="strlist">Volume</append>
<append key="Volume.method_names" type="strlist">Eject</append>
<append key="Volume.method_signatures" type="strlist">as</append>
<append key="Device.Volume.method_execpaths" type="strlist">hal-storage-
eject</append>

</match>
```

After dealing with special cases, the file system devices are matched as shown here:

```
<match key="volume.fsusage" string="filesystem">
```

Storage devices to ignore like vfat are specified.

```
<merge key="volume.ignore" type="bool">false</merge>
```

Then the actions to take and the callout script to use are specified such as the one for Unmount that uses **hal-storage-mount**.

```
<append key="Device.Volume.method_names" type="strlist">Mount</append>
<append key="Device.Volume.method_signatures" type="strlist">ssas</append>
<append key="Device.Volume.method_execpaths" type="strlist">hal-storage-
mount</append>
```

Options are then specified with Volume.mount.valid_options, starting with defaults and continuing with special cases, like **ext3** shown here.

```
<!-- allow these mount options for ext3 -->

<match key="volume.fstype" string="ext3">
<append key="volume.mount.valid_options"type="strlist">data=</append>
 </match>
```

HAL Callouts

Callouts are programs invoked when the device object list is modified or when a device changes. As such, callouts can be used to maintain system-wide policy (that may be specific to the particular OS) such as changing permissions on device nodes, managing removable devices, or configuring the networking subsystem. There are three different kinds of callouts for devices, capabilities, and properties. *Device* callouts are run when a device is added or removed. *Capability* callouts add or remove device capabilities, and *property* callouts add or remove a device's property. Callouts are implemented using **method** property rules, such as the one that invokes the **hal-storage-mount** callout when CD/DVD-ROMs are inserted or removed as shown here:

```
<append key="org.freedesktop.Hal.Device.Volume.method_execpaths"
type="strlist">hal-storage-mount</append>
```

Callouts are placed in the **/usr/libexec** directory with the HAL callouts prefixed with **hal-**. . Here you will find many of storage callouts used by HAL such as **hal-storage-eject** and **hal-storage-mount**. The **gnome-mount** tool used for mounting CD/DVD disk on the Gnome desktop uses the HAL callouts.

Manual Devices

Several devices still need to be created manually, printer parallel ports, for example. Most of these devices are already configured with MAKEDEV and the **/etc/makedev.d** files. To have these devices created by udev, their names are placed in configuration files in the **/etc/udev/makedev.d** directory. The **50-udev.nodes** file contains a list of device names that udev will use MAKEDEV to manually construct when udev generates the **/dev** device directory. Here you will find entries for parallel ports like **paraport0** through **parport3**.

You can, if you wish, create device file interfaces manually yourself using the **MAKEDEV** or **mknod** commands. To have them added to the **/dev** directory by udev, place them in the **/etc/udev/devices** directory; udev will copy them for you to the **/dev** directory when it generates them. For some devices, such as ISDN devices, you may have to do this. The following example makes an ISDN device using **MAKEDEV** and places it in the **/etc/udev/devices** directory.

```
/sbin/MAKEDEV -d /etc/udev/devices isdn
```

Device Types

Linux implements several types of devices, the most common of which are block and character. A *block device,* such as a hard disk, transmits data a block at a time. A *character device,* such as a printer or modem, transmits data one character at a time, or rather as a continuous stream of data, not as separate blocks. Device driver files for character devices have a *c* as the first character in the permissions segment displayed by the **ls** command. Device driver files for block devices have a *b.* In the next example, **lp0** (the printer) is a character device and **sda1** (the hard disk) is a block device:

```
# ls -l sda1 lp0
brw-rw---- 1 root disk 3, 1 Jan 30 02:04 sda1
crw-rw---- 1 root lp   6, 0 Jan 30 02:04 lp0
```

The device type can be either *b, c, p,* or *u.* As already mentioned, the *b* indicates a block device, and *c* is for a character device. The *u* is for an unbuffered character device, and the *p* is for a FIFO (first in, first out) device. Devices of the same type often have the same name; for example, serial interfaces all have the name **ttyS**. Devices of the same type are then uniquely identified by a number attached to the name. This number has two components: the major number and the minor number. Devices may have the same major number, but if so, the minor number is always different. This major and minor structure is designed to deal with situations in which several devices may be dependent on one larger device, such as several modems connected to the same I/O card. All the modems would have the same major number that references the card, but each modem would have a unique minor number. Both the minor and major numbers are required for block and character devices (*b, c,* and *u*). They are not used for FIFO devices, however.

Valid device names along with their major and minor numbers are listed in the **devices.txt** file located in the **/Documentation** directory for the kernel source code, **/usr/src/linux-***ver***/Documentation** (also at www.kernel.org/pub/linux/docs/device-list/devices.txt). When you create a device, you use the major and minor numbers as well as the device name prefix for the particular kind of device you are creating. Most of these devices are already created for you and are listed in the **/dev** directory.

MAKEDEV

You use **MAKEDEV** to create device files. **MAKEDEV** uses device configuration files located in the **/etc/makedev.d** directory to determine device options like the major or minor number of the device or any symbolic links that should be created for it. For example, the **/etc/makedev.d/01sound** file lists sound devices. A **MAKEDEV** configuration file can have three different kinds of records, each beginning with a different operator:

- **b or c** Create a block (b) or character (c) device. These entries hold several options: mode (permissions), owner, group, major and minor numbers, inc, count (number of devices created), and fmt (the name of the device). The fmt option is technically a format string, which can include a format specifier for numerically incrementing names of the similar devices, such as **cdrom%d** for **cdrom0**, **cdrom1**, and so on. The inc option sets an increment.
- **l** Creates a symbolic link for a device.
- **a** An alias applies the commands used for one device for those of another. This lets you create a sound device, which in turn automatically creates audio, midi, and mixer devices, and so on.

In the **/etc/makedev.d/01sound** file, there are numerous alias entries for sound, such as the following:

```
a  sound audio
```

A link entry will create a symbolic link called **audio0** for the audio device file.

```
l audio0 audio
```

The actual sound device file creation is configured in the **alsa** file. Sound devices use ALSA sound drivers. Here you will find numerous c entries with permission, owner, group values, etc.

With so much of the configuration handled in the **MAKEDEV** device configuration files, the command to create a device is very simple. However, bear in mind that with udev, device files cannot be created in **/dev**. This directory is automatically regenerated by udev. To have udev place your device file in **/dev** when it generates it, you place the device file you made in **/etc/udev/devices**. Use the **-d** option to specify the udev device directory. The following would create an ISDN device:

```
MAKEDEV -d /etc/udev/devices  isdn
```

mknod

Though the **MAKEDEV** command is preferable for creating device files, it can only create files for which it is configured. For devices not configured for use by **MAKEDEV**, you will have to use the **mknod** command. This is a lower-level command that requires manual configuration of all its settings. With the **mknod** command you can create a device file in the traditional manner without any of the configuration support that **MAKEDEV** provides.

The **mknod** command can create either a character or block-type device. The **mknod** command has the following syntax:

```
mknod options device device-type major-num minor-num
```

As most devices are easily covered by **MAKEDEV** as well as automatically generated by udev, you will rarely if ever need to use **mknod**. As a simple example, creating a device file with **mknod** for a printer port is discussed here. Linux systems usually provide device files for printer ports (**lp0**–**2**). As an example, you can see how an additional port could be created manually with the **mknod** command. Printer devices are character devices and must be owned by the root and daemon. The permissions for printer devices are read and write for the owner and the group, 660. The major device number is set to 6, while the minor device number is set to the port number of the printer, such as 0 for LPT1 and 1 for LPT2. Once the device is created, you use **chown** to change its ownership to the **root** user, since only the administrator should control it. Change the group to **lp** with the **chgrp** command.

Most devices belong to their own groups, such as **disks** for hard disk partitions, **lp** for printers, **floppy** for floppy disks, and **tty** for terminals. In the next example, a printer device is made on a fourth parallel port, **lp3**. The **-m** option specifies the permissions—in this case, 660. The device is a character device, as indicated by the **c** argument following the device name. The major number is 6, and the minor number is 3. If you were making a device at **lp4**, the major number would still be 6, but the minor number would be 4. Once the device is made, the **chown** command then changes the ownership of the parallel printer device to **root**. For printers, be sure that a spool directory has been created for your device. If not, you need to make one. Spool directories contain files for data that varies according to the device output or input, like that for printers or scanners.

As with all manual devices, the device file has to be placed in the **/etc/udev/devices** directory; udev will later put it in **/dev**.

```
# mknod -m 660 /etc/udev/devices/lp3 c 6 3
# chown root /etc/udev/devices/lp3
# chgrp lp /etc/udev/devices/lp3
```

Installing and Managing Terminals and Modems

In Linux, several users may be logged in at the same time. Each user needs his or her own terminal through which to access the Linux system, of course. The monitor on your PC acts as a special terminal, called the *console,* but you can add other terminals through either the serial ports on your PC or a special multiport card installed on your PC. The other terminals can be stand-alone terminals or PCs using terminal emulation programs. For a detailed explanation of terminal installation, see the **Term-HOWTO** file in **/usr/share/doc/HOWTO** or at the Linux Documentation Project site (**http://tldp.org**). A brief explanation is provided here.

Serial Ports

The serial ports on your PC are referred to as COM1, COM2, COM3, and COM4. These serial ports correspond to the terminal devices **/dev/ttyS0** through **/dev/ttyS3**. Note that several of these serial devices may already be used for other input devices such as your mouse, and for communications devices such as your modem. If you have a serial printer, one of these serial devices is already used for that. If you installed a multiport card, you have many more ports from which to choose. For each terminal you add, udev will create the appropriate character device on your Linux system. The permissions for a terminal device are normally 660. *Terminal devices* are character devices with a major number of 4 and minor numbers usually beginning at 64.

Note: Terminal devices are managed by your system using the **`mingetty`** program. When your system starts, it sets up a set of connected terminals using TTY upstart files in the /etc/event.d directory, like tty1. These runs the mingetty program with the terminal device name, like **tty1**.

terminfo and /etc/event.d Files

The **/etc/inittab** file is no longer used to hold instructions for your system on how to manage terminal devices. Instead separate files in the **/etc/event.d** directory are uses, one for each terminal. The files are named with the **tty** prefix, like **tty1**.

The files have four basic components: an ID, a runlevel, an action, and a process. Terminal devices are identified by ID numbers, beginning with 1 for the first device. The runlevel at which the terminal operates is usually 1. The action is usually *respawn,* which means to run the process continually. The process is a call to the **`mingetty`**, **`mgetty`**, or **`agetty`** with the terminal device name.

The **/etc/terminfo** and **/usr/share/terminfo** directories holds the specification files for different terminal types. These are the different types of terminals users could use to log in to your system. It replaces the older **/etc/termcap** file. See the terminfo man page for more details.

tset

When a user logs in, having the terminal device initialized using the **`tset`** command is helpful. Usually the **`tset`** command is placed in the user's **.bash_profile** file and is automatically executed whenever the user logs in to the system. You use the **`tset`** command to set the terminal type and any other options the terminal device requires. A common entry of **`tset`** for a **.bash_profile** file follows. The **`-m dialup:`** option prompts the user to enter a terminal type. The type specified here is a default type that is displayed in parentheses. The user presses ENTER to choose the default. The prompt looks like this: **`TERM=(vt100)?`**.

```
eval 'tset -s -Q -m dialup:?vt00'
```

Input Devices

Input devices, such as mice and keyboards, are displayed on several levels. Initial detection is performed during installation where you select the mouse and keyboard types. Keyboard and mice will automatically be detected by HAL. You can perform detailed configuration with your desktop configuration tools, such as the Gnome or KDE mouse

configuration tools. On Gnome, select System | Preferences | Hardware | Mouse to configure your mouse. There is a Keyboard entry on that same menu for keyboards.

Installing Sound, Network, and Other Cards

Support for most devices is provided in the form of kernel modules that can be dynamically loaded into the kernel. For example, drivers for an AC97 sound card are in the module **snd-ac97-codec.ko**. Loading this module makes your sound card accessible to Linux. Most Linux distributions automatically detect the cards installed on your system and load the needed modules. Sound, network, and other devices are installed automatically by HAL and udev. The needed kernel module is selected and loaded for you. If you change sound cards, the new card is automatically detected. You could also load modules you need manually, removing an older conflicting one. Installing support for a card is usually a simple matter of loading a module that includes the drivers for it.

Device files for most devices are already set up for you in the **/dev** directory by **udev**. For example, the device name for your sound card is **/dev/audio**.

Sound and Video Devices

Sound devices are now automatically detected and configured by HAL and udev (system-config-soundcard used in previous releases has been dropped). Some sound cards may require more specialized support.

For the 2.4 kernel, most Linux sound drivers were developed as part of the Open Sound System (OSS) and freely distributed as OSS/Free. These are installed as part of Linux distributions. The OSS device drivers are intended to provide a uniform API for all Unix platforms, including Linux. They support Sound Blaster– and Windows Sound System–compatible sound cards (ISA and PCI).

The Advanced Linux Sound Architecture (ALSA) replaced OSS in the 2.6 Linux kernel; it aims to be a better alternative to OSS, while maintaining compatibility with it. ALSA provides a modular sound driver, an API, and a configuration manager. ALSA is a GNU project and is entirely free; its Web site at **www.alsa-project.org** contains extensive documentation, applications, and drivers. Currently available are the ALSA sound driver, the ALSA Kernel API, the ALSA library to support application development, and the ALSA manager to provide a configuration interface for the driver. ALSA evolved from the Linux Ultra Sound Project.

A video device will have the name **/dev/video**. Supporting kernel modules are loaded automatically. Digital Video Broadcast (DVB) devices like TV tuner cards are located in the **/dev/dvb** directory. Device names begin with **adapter**, starting with **adapter0**.

PCMCIA Devices

PCMCIA devices are card readers commonly found on laptops to connect devices like modems or wireless cards, though they are becoming standard on many desktop systems as well. The same PCMCIA device can support many different kinds of devices, including network cards, modems, hard disks, and Bluetooth devices.

PCMCIA support is now managed by udev and HAL. You no longer use the **cardmgr/pcmcia** service. PCMCIA devices are now considered hotplugged devices managed by

HAL and udev directly. Card information and control is now managed by **pccardctl**. The PCMCIA udev rules are listed in **/etc/udev/rules.d/60-pcmcia.rules**, which automatically probes and install cards. Check www.kernel.org/pub/linux/utils/kernel/pcmcia/pcmcia.html for more information.

You can obtain information about a PCMCIA device with the **pccardctl** command, as well as manually eject and insert a device. The **status**, **config**, and **ident** options will display the device's socket status and configuration, and the identification of the device. The **insert** and **eject** options will let you add and remove a device. The **cardinfo** command also provides device information.

It is not advisable to hot-swap IDE or SCSI devices. For these you should first manually shut down the device using the **pccardctl** command.

```
pccardctl eject
pccardctl scheme home
```

Modules

The Linux kernel employs the use of modules to support different operating system features, including support for various devices such as sound and network cards. In many cases, you do have the option of implementing support for a device either as a module or by directly compiling it as a built-in kernel feature, which requires you to rebuild the kernel. A safer and more robust solution is to use modules. *Modules* are components of the Linux kernel that can be loaded as needed. To add support for a new device, you can now simply instruct a kernel to load the module for that device. In some cases, you may have to recompile only that module to provide support for your device. The use of modules has the added advantage of reducing the size of the kernel program as well as making your system more stable. The kernel can load modules in memory only as they are needed. Should a module fail, only the module stops running, and it will not affect the entire system. For example, the module for the PPP network interface used for a modem needs to be used only when you connect to an ISP.

Kernel Module Tools

The modules your system needs are usually determined during installation, according to the kind of configuration information you provided and the automatic detection performed by your Linux distribution. For example, if your system uses an Ethernet card whose type you specified during installation, the system loads the module for that card. You can, however, manually control what modules are to be loaded for your system. In effect, this enables you to customize your kernel whatever way you want. You can use several commands, configuration tools, and daemons to manage kernel modules. The 2.6 Linux kernel includes the Kernel Module Loader (Kmod), which has the capability to load modules automatically as they are needed. Kernel module loading support must also be enabled in the kernel, though this is usually considered part of a standard configuration. In addition, several tools enable you to load and unload modules manually, if you must. The Kernel Module Loader uses certain kernel commands to perform the task of loading or unloading modules. The **modprobe** command is a general-purpose command that calls **insmod** to load modules and **rmmod** to unload them. These commands are listed in Table 17-9.

Options for particular modules, general configuration, and even specific module loading can be specified in the **/etc/modprobe.conf** file. You can use this file to automatically load and configure modules. You can also specify modules to be loaded at the boot prompt or in **grub.conf**.

Command	Description
lsmod	Lists modules currently loaded.
insmod	Loads a module into the kernel. Does not check for dependencies.
rmmod	Unloads a module currently loaded. Does not check for dependencies.
modinfo	Displays information about a module: **-a** (author),**-d** (description), **-p** (module parameters), **-f** (module filename), **-v** (module version).
depmod	Creates a dependency file listing all other modules on which the specified module may rely.
modprobe	Loads a module with any dependent modules it may also need. Uses the file of dependency listings generated by **depmod**: **-r** (unload a module), **-l** (list modules).

Table 17-9: Kernel Module Commands

Module Files and Directories: /lib/modules

The filename for a kernel module has the extension **.ko**. Kernel modules reside in the **/lib/modules/***version* directory, where *version* is the version number for your current kernel with the extension **fc10** and the processor type, x86_64 or i686. The directory for the **2.6.27.5-159.fc10.x86_64** kernel is **/lib/modules/2.6.27.5-159.fc10.x86_64**. As you install new kernels on your system, new module directories are generated for them. One method to access the directory for the current kernel is to use the **uname -r** command to generate the kernel version number. This command needs to have backquotes.

```
cd /lib/modules/`uname -r`
```

In this directory, modules for the kernel reside in the **/kernel** directory. Within the **/kernel** directory are several subdirectories, including the **drivers** directory that holds subdirectories for modules like network and video drivers (net and video subdirectories). These subdirectories serve to categorize your modules, making them easier to locate. Sound modules are held separately in the **kernel/sound** directory.

Managing Modules with /etc/moprobe.d

As noted previously, there are several commands you can use to manage modules. The **lsmod** command lists the modules currently loaded into your kernel, and **modinfo** provides information about particular modules. Though you can use the **insmod** and **rmmod** commands to load or unload modules directly, you should use only **modprobe** for these tasks. Often, however, a given module requires other modules to be loaded. For example, the module for an AC97 sound card, **snd-ac97-codec.ko**, requires the **soundcore.ko** module to be loaded also.

The depmod Command

Instead of manually trying to determine what modules a given module depends on, you use the **depmod** command to detect the dependencies for you. The **depmod** command generates a file that lists all the modules on which a given module depends. The **depmod** command generates a

hierarchical listing, noting what modules should be loaded first and in what order. Then, to load the module, you use the **modprobe** command using that file. **modprobe** reads the file generated by **depmod** and loads any dependent modules in the correct order, along with the module you want. You need to execute **depmod** with the **-a** option once, before you ever use **modprobe**. Entering **depmod -a** creates a complete listing of all module dependencies. This command creates a file called **modules.dep** in the module directory for your current kernel version, **/lib/modules/***version*.

```
depmod -a
```

The modprobe Command

To install a module manually, you use the **modprobe** command and the module name. You can add any parameters the module may require. The following command installs the Intel high definition sound module. **modprobe** also supports the use of the * character to enable you to use a pattern to select several modules. This example uses several values commonly used for sound cards. You would use the values recommended for your sound card on your system. Most sound card drivers are supported by the ALSA project. Check their Web site for find what driver module is used for your card.

```
modprobe  snd-hda-intel
```

To discover what parameters a module takes, you can use the **modinfo** command with the **-p** option.

You can use the **-l** option to list modules and the **-t** option to look for modules in a specified subdirectory. Sound modules are arranged in different subdirectories according to the device interface they use, like **pci**, **isa**, or **usb**. Most internal sound cards use **pci**. Within the interface directory, there may be further directories like emu10k1 used for the Creative Audigy cards and hda for high definition drivers. In the next example, the user lists all modules in the **sound/pci/hda** directory:

```
# modprobe -l -t sound/pci/hda
```

Options for the **modprobe** command are placed in files located the **/etc/modprobe.d** directory. Here, you can enter configuration options, such as default directories and aliases. An alias provides a simple name for a module.

The insmod Command

The **insmod** command performs the actual loading of modules. Both **modprobe** and the Kernel Module Loader make use of this command to load modules. Though **modprobe** is preferred, because it checks for dependencies, you can load or unload particular modules individually with **insmod** and **rmmod** commands. The **insmod** command takes as its argument the name of the module, as does **rmmod**. The name can be the simple base name, like **snd-ac97-codec** for the **snd-ac97-codec.ko** module. You can specify the complete module filename using the **-o** option. Other helpful options are the **-p** option, which lets you probe your system first to see if the module can be successfully loaded, and the **-n** option, which performs all tasks except actually loading the module (a dummy run). The **-v** option (verbose) lists all actions taken as they occur. In those rare cases where you may have to force a module to load, you can use the **-f** option. In the next example, **insmod** loads the **snd-ac97-codec.ko** module:

```
# insmod -v snd-ac97-codec
```

The rmmod Command

The **rmmod** command performs the actual unloading of modules. It is the command used by **modprobe** and the Kernel Module Loader to unload modules. You can use the **rmmod** command to remove a particular module as long as it is not being used or required by other modules. You can remove a module and all its dependent modules by using the **-r** option. The **-a** option removes all unused modules. With the **-e** option, when **rmmod** unloads a module, it saves any persistent data (parameters) in the persistent data directory, usually **/var/lib/modules/persist**.

The /etc/modprobe.d Directory

Module loading can require system renaming as well as specifying options to use when loading specific modules. Even when removing or installing a module, certain additional programs may have to be run or other options specified. These parameters can be set in module configuration files in the **/etc/modprobe.d** directory. Configuration for modporbe supports several actions including alias, options, install, and remove.

- **alias** *module name* Provides another name for the module, used for network and sound devices.
- **options** *module options* Specifies any options a particular module may need.
- **install** *module commands* Uses the specified commands to install a module, letting you control module loading.
- **remove** *module commands* Specifies commands to be run when a module is unloaded.
- **include** *config-file* Additional configuration files.
- **blacklist** *module* Ignore any internal aliases that given module may define for itself.

  This allows you to use only aliases defined by modprobe.

Tip: After making changes to **/etc/modprobe.conf**, you should run **depmod** again to record any changes in module dependencies.

Note: Instead of a single **modprobe.conf** file, modprobe configuration is implemented using separate files in an **/etc/modprobe.d** directory.

Most devices are handled by udev and HAL. If you need to load or configure a particular device, you can set up a **modprobe.d** device file for it. Nvidia will set one up for its Linux graphics driver.

Tip: In some cases, Kmod may not detect a device in the way you want, and thereby not load the kernel module you would like. In this case, kernel parameters were specified to the GRUB boot loader to load the correct modules.

Installing New Modules from Vendors: Driver Packages

Often you may find that your hardware device is not supported by current Linux modules. In this case you may have to download drivers from the hardware vendor or open source development group to create your own driver and install it for use by your kernel.

The drivers could be in RPM or compressed archives. The process for installing drivers differs, depending on how a vendor supports the driver. Different kinds of packages are listed here:

- **RPM or Deb packages** Some support sites will provide drivers already packaged in RPM or Deb files for direct installation.
- **Drivers compiled in archives** Some will provide drivers already compiled for your distribution, but packaged in compressed archives. In this case a simple install operation will place the supporting module in the **modules** directory and make if available for use by the kernel.
- **Source code** Others provide just the source code, which, when compiled, will detect your system configuration and compile the module accordingly.
- **Scripts with source code** Some will provide customized scripts that may prompt you for basic questions about your system and then both compile and install the module.

For drivers that come in the form of compressed archives (**tar.gz** or **tar.bz2**), the compile and install operations normally make use a Makefile script operated by the **make** command. A simple install would usually just require running the following command in the driver's software directory:

```
make install
```

In the case of sites that just supply the source code, you may have to perform both configure and compile operations as you would for any software.

```
./configure
make
make install
```

For packages that have no install option, compiled or source, you will have to manually move the module to the kernel module directory, **/lib/modules**/*version,* and use **depmod** and **modprobe** to load it (see the preceding section).

If a site gives you a customized script, you would just run that script. For example, the Marvel gigabit LAN network interfaces found on many motherboards use the SysKonnect Linux drivers held in the **skge.ko** module. The standard kernel configuration will generate and install this module. But if you are using a newer motherboard, you may need to download and install the latest Linux driver. For example, some vendors may provide a script, **install.sh** that you run to configure, compile, and install the module.

```
./install.sh
```

Note: On Red hat and Fedora, if you are only provided a source code file for the module, such as a **.c** file, you can use the kernel header files in the **/lib/modules**/*version*/**build** directory to compile the module. See the Fedora or Red Hat Release Note for details on how to create a customized Makefile for creating modules. You will not have to download and install the source code.

Installing New Modules from the Kernel

The source code for your Linux kernel contains an extensive set of modules, of which not all are actually used on your system. The kernel binaries provided by most distributions come with a extensive set of modules already installed. If, however, you install a device for which kernel support is not already installed, you will have to configure and compile the kernel module that provides the drivers for it. This involves using the kernel source code to select the module you need from a list in a kernel configuration tool, and then regenerating your kernel modules with the new module included. Then the new module is copied into the module library, installing it on your system. You can also enter it a file in the **/etc/modprobe.d** directory with any options, or use **modprobe** to install it manually.

Download the Fedora source code version of the kernel from the fedora-source repository (see Chapter 7). Install to a local user directory, setting up a local build directories using **rpmdev-setuptree**, and then extract the kernel for your system. Use **`uname -m`** to specify the currently installed kernel version on your system.

```
rpmbuild -bp --target=`uname -m` kernel.spec
```

Change to the **BUILD** directory in your local kernel directory. Then use the **make** command with the **gconfig** or **menuconfig** argument to display the kernel configuration menus, invoking them with the following commands. The **make gconfig** command starts an X Window System interface that needs to be run on your desktop from a terminal window.

```
make gconfig
```

Using the menus select the modules you need. Make sure each is marked as a module, clicking the Module check box in **gconfig** or typing **m** for **menuconfig**. Once the kernel is configured, save it and exit from the configuration menus. Then you compile the modules, creating the module binary files with the following command:

```
make modules
```

This places the modules in the kernel source modules directory. You can copy the one you want to the kernel modules directory, **/lib/modules/***version***/kernel**, where *version* is the version number of your Linux kernel. A simpler approach is to reinstall all your modules, using the following command. This copies all the compiled modules to the **/lib/modules/***version***/kernel** directory:

```
make modules_install
```

fedora

18. Backup

Individual Backups: archive and rsync

BackupPC

Amanda

Backups with dump and restore

Backup operations have become an important part of administrative duties. Several backup tools are provided on Linux systems, including Anaconda and the traditional dump/restore tools, as well as the **rsync** command for making individual copies. Anaconda provides server-based backups, letting different systems on a network back up to a central server. BackupPC provides network and local backup using configured **rsync** and **tar** tools. The dump tools let you refine your backup process, detecting data changed since the last backup. Table 18-1 lists Web sites for Linux backup tools.

Web Site	Tools
http://rsync.samba.org	rsync remote copy backup
www.amanda.org	Amanda network backup
http://dump.sourceforge.net	dump and restore tools
http://backuppc.sourceforge.net	BackupPC network or local backup using configured **rsync** and **tar** tools.

Table 18-1: Backup resources

Individual Backups: archive and rsync

You can back up and restore particular files and directories with archive tools like **tar**, restoring the archives later. For backups, **tar** is usually used with a tape device. To automatically schedule backups, you can schedule appropriate **tar** commands with the **cron** utility. The archives can be also compressed for storage savings. You can then copy the compressed archives to any medium, such as a DVD disc, a floppy, or tape. On GNOME you can use File Roller to create archives easily (Archive Manager under System Tools). The Kdat tool, a front end to **tar**, on KDE will back up to tapes.

If you want to remote-copy a directory or files from one host to another, making a particular backup, you can use **rsync**, which is designed for network backups of particular directories or files, intelligently copying only those files that have been changed, rather than the contents of an entire directory. In archive mode, it can preserve the original ownership and permissions, providing corresponding users exist on the host system. The following example copies the **/home/george/myproject** directory to the **/backup** directory on the host **rabbit**, creating a corresponding **myproject** subdirectory. The **-t** specifies that this is a transfer. The remote host is referenced with an attached colon, **rabbit:**.

```
rsync -t /home/george/myproject  rabbit:/backup
```

If, instead, you wanted to preserve the ownership and permissions of the files, you would use the **-a** (archive) option. Adding a **-z** option will compress the file. The **-v** option provides a verbose mode.

```
rsync -avz  /home/george/myproject  rabbit:/backup
```

A trailing slash on the source will copy the contents of the directory, rather than generating a subdirectory of that name. Here the contents of the **myproject** directory are copied to the **george-project** directory.

```
rsync -avz  /home/george/myproject/   rabbit:/backup/george-project
```

The **rsync** command is configured to use SSH remote shell by default. You can specify it or an alternate remote shell to use with the **-e** option. For secure transmission you can encrypt the copy operation with ssh. Either use the **-e ssh** option or set the **RSYNC_RSH** variable to ssh.

```
rsync -avz -e ssh  /home/george/myproject   rabbit:/backup/myproject
```

As when using **rcp**, you can copy from a remote host to the one you are on.

```
rsync -avz  lizard:/home/mark/mypics/  /pic-archice/markpics
```

You can also run rsync as a server daemon. This will allow remote users to sync copies of files on your system with versions on their own, transferring only changed files rather than entire directories. Many mirror and software FTP sites operate as rsync servers, letting you update files without have to download the full versions again. Configuration information for rsync as a server is kept in the **/etc/rsyncd.conf** file. On Linux, rsync as a server is managed through **xinetd**, using the **/etc/xinetd.d/rsync** file, which starts **rsync** with the **--daemon** option. In the **/etc/services** file, it is listed to run on port 873. It is off by default, but you can enable it with at tool like **chkconfig** or system-config-services.

Tip: Though it is designed for copying between hosts, you can also use **rsync** to make copies within your own system, usually to a directory in another partition or hard drive. In fact there are eight different ways of using **rsync**. Check the **rsync** Man page for detailed descriptions of each.

BackupPC

BackupPC provides an easily managed local or network backup of your system or hosts on a system using configured rsync or tar tools. There is no client application to install, just configuration files. BackupPC can backup hosts on a network, including servers, or just a single system. Data can be backed up to local hard disks or to network storage like shared partitions or storage servers. Detailed documentation is installed at **/usr/share/doc/BackupPC**. You can find out more about BackupPC at **http://backuppc.sourceforge.net**.

BackupPC uses both compression and detection of identical files to significantly reduce the size of the backup, allowing several hosts to be backed up in limited space. Once an initial backup is performed, BackupPC will only backup changed files, reducing the time of the backup significantly.

You can configure BackupPC using your Web page configuration interface. This is the host name of your computer with the **/BackupPC** name attached, like **http://rabbit/BackupPC**. You can also use localhost, **http://localhost/BackupPC**. Detailed documentation is installed at **/usr/share/doc/BackupPC**. Configuration files are located at **/etc/BackupPC**. The **config.pl** file holds BackupPC configuration options and the **hosts** file lists hosts to be backed up. Keep in mind that capital letters are used in the name.

```
BackupPC
```

You can use **system-config-services** to have BackupPC start automatically. Check the backuppc entry. BackupPC has its own service script with which you start, stop, and restart the BackupPC service manually, **/etc/init.d/backuppc**.

```
service backuppc start
```

On Fedora, you will first have to set up your configuration manually. Check the **/usr/share/doc/backuppc/fedora.README** file for instructions.

- Set up a user name and password for accessing your BackupPC Web configuration interface using the **htpassword** command and the **/etc/BackupPC/apache.users** file.
- Be sure that both the httpd and backuppc servers are then running.
- Add host entries in the **/etc/BackupPC/hosts** file for hosts you want to backup.
- Access the BackupPC configuration interface with the URL **http://localhost/BackupPC** on your Web browser.

You first set up a user name and password using the htpasswd command. User names and passwords are placed in the **/etc/BackupPC/apache.users** file. For security reasons you cannot login as the backuppc user. Specify a new user and a password for it. The **apache.users** file will not initially exist. The first time you use this operation, add a **-c** option to create the **apache.users** file. Login as the root user first, **su** command.

```
htpassword -c /etc/BackupPC/apache.users  richard
```

You will be prompted to enter a password, and repeat it.

You can then able to access and login to the BackupPC Web configuration interface.

You then should add entries in the **/etc/BackupPC/hosts** file for the hosts you want to backup. Sample entries are listed for both DHCP and static IP addressed hosts. An entry consists of the name of the host, the DHCP flag, and the users that can access the BackupPC interface for that host. The DHCP flag of 1 indicates a DHCP host, whereas a 0 is used for static hosts. The following entry sets up access for localhost as a DHCP address, and accessible by the BackupPC **richard** user.

```
locahost 1 richard
```

First, make sure that both the **httpd** (Apache Web server) and the **backuppc** servers are running. You can use system-config-services to enable and start these servers.

```
service httpd start
service backuppc start
```

To access BackupPC, start your browser and enter the BackupPC URL (computer name with /BackupPC). You can also use localhost instead of the name of your computer. A dialog will open with entries for the user name and password. Enter the user name and password you set up for BackupPC.

```
http:/localhost/BackupPC
```

The general welcome and status screen is displayed (see Figure 18-1). The left sidebar has three sections, one for your current host, another for selecting hosts titled Hosts, and another for the BackupPC server which is titled Server.

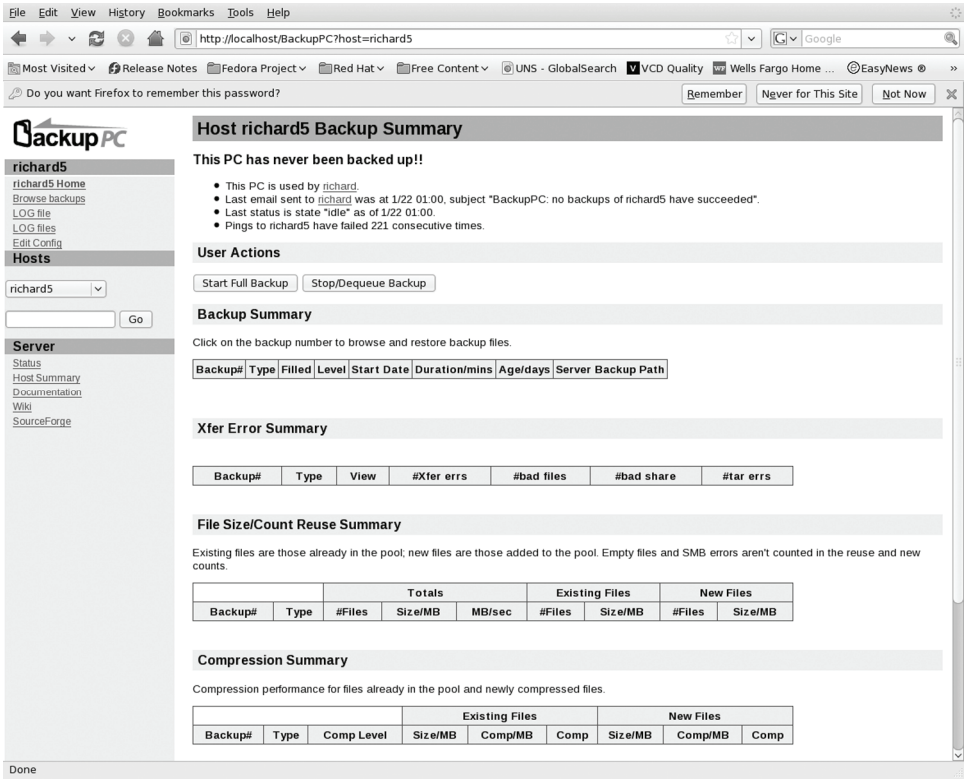


Figure 18-1: BackupPC host page

BackupPC host backup and configuration

The Hosts pop-up menu is located on the sidebar in the Hosts section. Here you choose the host to perform backup and restores on. The localhost entry will access your own computer. When you select a host, a new section will appear on the sidebar above the Host section, labeled with that host name, like localhost. In this section will be links for the host home page, Browse backups, logs, and an Edit Config link to configure the backup for that host,

The host home page will list backups and display buttons for full and incremental backups (see Figure 18-1). Click on the Start Full Backup to perform a full backup. Once you have performed your first backup, you will also have the option to do incremental backups. Click the Start Incre Backup for an incremental backup (changed data only). You will be prompted for confirmation before the backup begins.

To select files to restore, you click the Browse backups link to display a tree of possible files and directories to restore. Select the ones you want or just click the Select All checkbox choose the entire backup. Then click on Restore Backup. A Restore page then lets you choose from three kinds of backup: a direct restore, Zip archive, or tar archive. For a direct restore you can have BackupPC either overwrite your current files with the restored ones, or save them to a specified directory where you can later choose which ones to use. The Zip and Tar restore options create archive files that hold your backup. You can then later extract and restore files form the archive.

The Edit Config link open a page of tabbed panels for your host backup configuration. On the Xfer panel you can decide on the type of backup you want to perform. You can choose from archive (zip), tar, rsync, rsyncd, and smb (Samba). Here you can set specific settings like the destination directory for a zip archive or the Samba share to access for an smb backup. The Schedule panel is where you specify the backup intervals for full and incremental backups.

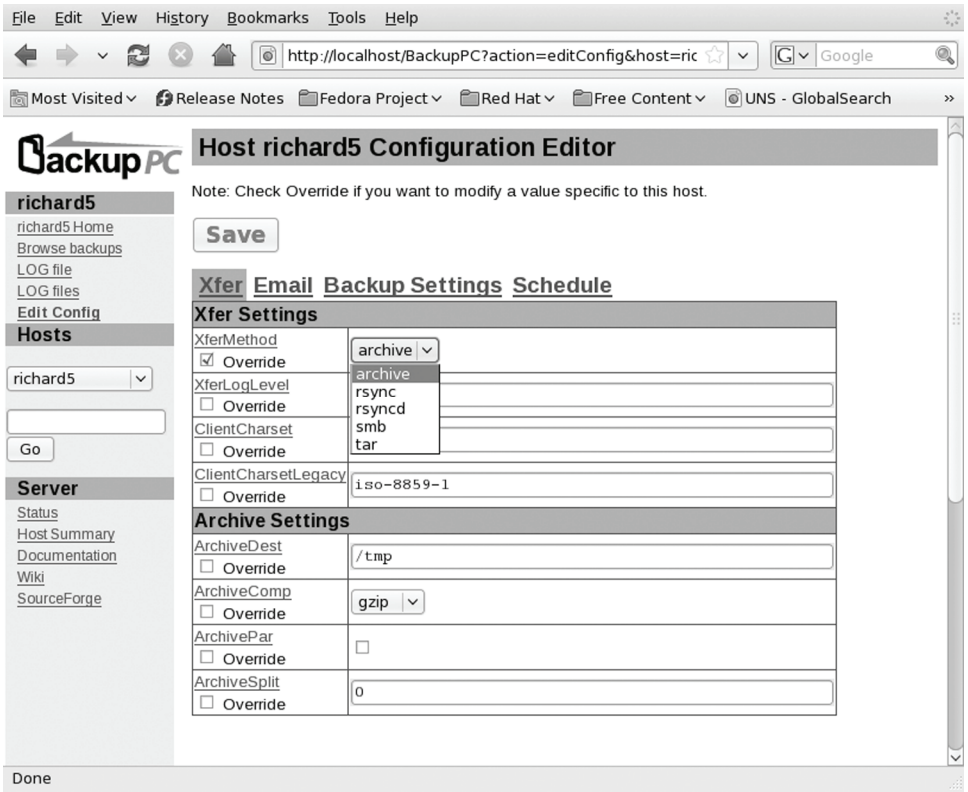


Figure 18-2: BackupPC Server Host Summary page

BackupPC server configuration

For security purposes on Fedora, you cannot login as the backuppc user with server configuration capability. To configure your server you would login as root (**su** command) and edit the **/etc/BackupPC/config.pl** configuration file.

Amanda

To back up hosts connected to a network, you can use the Advanced Maryland Automatic Network Disk Archiver (Amanda) to archive hosts. Amanda uses **`tar`** tools to back up all hosts to a single host operating as a backup server. Backup data is sent by each host to the host operating as the Amanda server, where they are written out to a backup medium such as tape. With an Amanda

server, the backup operations for all hosts become centralized in one server, instead of each host having to perform its own backup. Any host that needs to restore data simply requests it from the Amanda server, specifying the file system, date, and filenames. Backup data are copied to the server's holding disk and from there to tapes. Detailed documentation and updates are provided at www.amanda.org. For the server, be sure to install the amanda-server package, and for clients you would use the amanda-clients package.

Amanda is designed for automatic backups of hosts that may have very different configurations, as well as operating systems. You can back up any host that supports GNU tools, including Mac OS X and Windows systems connected through Samba.

Amanda Commands

Amanda has its own commands corresponding to the common backup tasks, beginning with "am," such as **amdump**, **amrestore**, and **amrecover**. The commands are listed in Table 18-2. The **amdump** command is the primary backup operation.

The **amdump** command performs requested backups; it is not designed for interactive use. For an interactive backup, you would use an archive tool like **tar** directly. The **amdump** is placed within a cron instruction to be run at a specified time. If, for some reason, **amdump** cannot save all its data to the backup medium (tape or disk), it will retain the data on the holding disk. The data can then later be directly written with the with the **amflush** command.

Command	Description
amdump	Perform automatic backups for the file systems listed in the disklist configuration file.
amflush	Used to directly back up data from the holding disk to a tape.
amcleanup	Clean up if there is a system failure on the server.
amrecover	Select backups to restore using an interactive shell.
amrestore	Restore backups, either files or complete systems.
amlabel	Label the backup medium for Amanda.
amcheck	Check the backup systems and files as well as the backup tapes before backup operations.
amadmin	Back up administrative tasks.
amtape	Manage backup tapes, loading and removing them.
amverify	Check format of tapes.
amverifyrun	Check the tapes from the previous run, specify the configuration directory for the backup.
amrmtape	Remove a tape from the Amanda database, use for damaged tapes.
amstatus	Show the status of the current Amanda backup operation.

Table 18-2: Amanda Commands

You can restore particular files as well as complete systems with the **amrestore** command. With the **amrecover** tool, you can select from a list of backups.

Amanda Configuration

Configuration files are placed in **/etc/amanda**, and log and database files in **/var/lib/amanda**. These are created automatically when you installed Amanda . You will also need to create a directory to use as a holding disk where backups are kept before writing to the tape. This should be on a file system with very large available space, enough to hold the backup of your largest entire host.

/etc/amanda

Within the **/etc/amanda** directory are subdirectories for the different kind of backups you want to perform. Each directory will contain its own **amanda.conf** and **disklist** file. By default a daily backup directory is created called **DailySet1**, with a default **amanda.conf** and a sample **disklist** file. To use them, you will have to edit them to enter your system's own settings. For a different backup configuration, you can create a new directory and copy the **DailySet1 amanda.conf** and **dislist** files to it, editing them as appropriate. When you issue Amanda commands like **amdump** to perform backups, you will use the name of the **/etc/amanda** subdirectory to indicate the kind of backup you want performed.

```
amdump DailySet1
```

The **/etc/amanda** directory also contains a sample cron file, **crontab.sample** that shows how a cron entry should look.

amanda.conf

The **amanda.conf** file contains basic configuration parameters such as the tape type and logfile as well as holding file locations. In most cases you can just use the defaults as listed in the **DailySet1/amanda.conf** file. The file is commented in detail, telling you what entries you will have to change. You will need to set the tapedev entries to the tape device you use, and the tape type entry for your tape drive type. In the holding disk segment, you will need to specify the partition and the directory for the holding disk you want to use. See the amanda Man page and documentation for detailed information on various options.

disklist

The **disklist** file is where you specify the file systems and partitions to be backed up. An entry lists the host, the partition, and the dump-type. The possible dump-types are defined in **amanda.conf**. The dump-types set certain parameters such as the priority of the backup and whether to use compression or not. The comp-root type will back up root partitions with compression and low priority, whereas the always-full type will back up an entire partition with no compression and the highest priority. You can define other dump-types in **amanda.conf** and use them for different partitions.

Backups will be performed in the order listed; be sure to list the more important ones first. The **disklist** file in **DailySet1** provides detailed examples.

Enabling Amanda on the Network

To use Amanda on the network, you need to run two servers on the Amanda server as well as an Amanda client on each network host. Access must be enabled for both the clients and the server.

Amanda Server

The Amanda server runs through **xinetd**, using **xinetd** service files located in **/etc/xinetd.d**. The two service files are **amidxtape** and **amandaidx**. Then restart the **xinetd** daemon to have it take immediate effect.

For clients to be able to recover backups from the server, the clients' host names must be placed in the **.amandahosts** file in the server's Amanda users' directory. This is **/var/lib/amanda**. On the server, **/var/lib/amanda/.amandahosts** will list all the hosts that are backed up by Amanda.

Amanda Hosts

Each host needs to allow access by the Amanda server. To do this, you place the host name of the Amanda server in each client's **.amandahosts** dot file. This file is located in the client's Amanda user home directory, **/var/lib/amanda**.

Each host needs to run the Amanda client daemon, **amanda**, which also runs under **xinetd**. Use **chkconfig** to turn it on.

```
chkconfig amanda on
```

Tip: If your server and hosts have firewalls, you will need to allow access through the ports that Amanda uses, usually 10080, 10082, and 10083.

Using Amanda

Backups are performed by the **amdump** command.

```
amdump DailySet1
```

An **amdump** command for each backup is placed in the Amanda **crontab** file. It is helpful to run an **amcheck** operation to make sure that a tape is ready.

```
0 16 * * 1-5 /usr/sbin/amcheck -m DailySet1
45 0 * * 2-6 /usr/sbin/amdump DailySet1
```

Before you can use a tape, you will have to label it with **amlabel**. Amanda uses the label to determine what tape should be used for a backup. Log in as the Amanda user (not root) and label the tape so that it can be used.

```
amlabel DailySet DailySet1
```

A client can recover a backup using **amrecover**. This needs to be run as the root user, not as the Amanda user. The **amrecover** command works through an interactive shell much like **ftp**, letting you list available files and select them to restore. Within the **amrecover** shell the **ls** command will list available backups, the **add** command will select one, and the extract operation will restore it. The **lcd** command lets you change the client directory; **amrecover** will use **DailySet1** as the default, but for other configurations you will need to specify their configuration

directory with the **-C** option. Should you have more than one Amanda server, you can list the one you want with the **-t** option.

```
amrecover -C DailySet1
```

To restore full system backups, you use the **amrestore** command, specifying the tape device and the host name.

```
amrestore  /dev/rmt1  rabbit
```

To select certain files, you can pipe the output to a recovery command such as **restore** (discussed in the next section).

```
amrestore -p /dev/rmt1  rabbit mydir  | restore  -ibvf 2 -
```

Backups with dump and restore

You can back up and restore your system with the dump and restore utilities. dump can back up your entire system or perform incremental backups, saving only those files that have changed since the last backup. dump supports several options for managing the backup operation, such as specifying the size and length of storage media (see Table 18-3).

The dump Levels

The dump utility uses *dump levels* to determine to what degree you want your system backed up. A dump level of 0 will copy file systems in their entirety. The remaining dump levels perform incremental backups, backing up only files and directories that have been created or modified since the last lower-level backup. A dump level of 1 will back up only files that have changed since the last level 0 backup. The dump level 2, in turn, will back up only files that have changed since the last level 1 backup (or 0 if there is no level 1), and so on up to dump level 9. You could run an initial complete backup at dump level 0 to back up your entire system, and then run incremental backups at certain later dates, having to back up only the changes since the full backup.

Using dump levels, you can devise certain strategies for backing up a file system. It is important to keep in mind that an incremental backup is run on changes from the last lower-level backup. For example, if the last backup was 6 and the next backup was 8, then the level 8 would back up everything from the level 6 backup. The sequence of the backups is important. If there were three backups with levels 3, then 6, and then 5, the level 5 backup would take everything from the level 3 backup, not stopping at level 6. Level 3 is the next-*lower*-level backup for level 5, in this case. This can make for some complex incremental backup strategies. For example, if you want each succeeding incremental backup to include all the changes from the preceding incremental backups, you could run the backups in descending dump level order. Given a backup sequence of 7, 6, and 5, with 0 as the initial full backup, 6 would include all the changes to 7, because its next lower level is 0. Then 5 would include all the changes for 7 and 6, also because its next lower level is 0, making all the changes since the level 0 full backup. A simpler way to implement this is to make the incremental levels all the same. Given an initial level of 0, and then two backups both with level 1, the last level 1 would include all the changes from the backup with level 0, since level 0 is the next *lower* level—not the previous level 1 backup.

Note: Several disk dump tools are also available. The **diskdumpfmt** command can be used to format tapes for use by dump. **diskdumpctl** registers a dump partition with the

system. `savecore` saves a **vmcore** file from the data in a dump partition. Dumped cores can be read by the **crash** tool. Check the **crash** Man page for details. .

Option	Description
`-0` through `-9`	Specifies the dump level. A dump level 0 is a full backup, copying the entire file system (see also the `-h` option). Dump level numbers above 0 perform incremental backups, copying all new or modified files new in the file system since the last backup at a lower level. The default level is 9.
`-B` *records*	Lets you specify the number of blocks in a volume, overriding the end-of-media detection or length and density calculations that dump normally uses for multivolume dumps.
`-a`	Lets dump bypass any tape length calculations and write until an end-of-media indication is detected. Recommended for most modern tape drives and is the default.
`-b` *blocksize*	Lets you specify the number of kilobytes per dump record. With this option, you can create larger blocks, speeding up backups.
`-d` *density*	Specifies the density for a tape in bits per inch (default is 1,600 BPI).
`-h` *level*	Files that are tagged with a user's nodump flag will not be backed up at or above this specified level. The default is 1, which will not back up the tagged files in incremental backups.
`-f` *file/device*	Backs up the file system to the specified file or device. This can be a file or tape drive. You can specify multiple filenames, separated by commas. A remote device or file can be referenced with a preceding hostname, *hostname*:*file.*
`-M` *file/device*	Implements a multivolume backup, where the *file* written to is treated as a prefix and the suffix consisting of a numbered sequence from 001 is used for each succeeding file, *file*001, *file* 002, etc. Useful when backup files need to be greater than the Linux **ext3** 2GB file size limit.
`-n`	Notifies operators if a backup needs operator attention.
`-s` *feet*	Specifies the length of a tape in feet. dump will prompt for a new tape when the length is reached.
`-S`	Estimates the amount of space needed to perform a backup.
`-T` *date*	Allows you to specify your own date instead of using the **/etc/dumpdates** file.
`-u`	Writes an entry for a successful update in the **/etc/dumpdates** file.
`-W`	Detects and displays the file systems that need to be backed up. This information is taken from the **/etc/dumpdates** and **/etc/fstab** files.
`-w`	Detects and displays the file systems that need to be backed up, drawing only on information in **/etc/fstab**.

Table 18-3: Options for dump

Recording Backups

Backups are recorded in the **/etc/dumpdates** file. This file will list all the previous backups, specifying the file system they were performed on, the dates they were performed, and the dump level used. You can use this information to restore files from a specified backup. Recall that the **/etc/fstab** file records the dump level as well as the recommended backup frequency for each file system. With the **-W** option, dump will analyze both the **/etc/dumpdates** and **/etc/fstab** files to determine which file systems need to be backed up. The **dump** command with the **-w** option just uses **/etc/fstab** to report the file systems ready for backup.

Operations with dump

The **dump** command takes as its arguments the dump level, the device it is storing the backup on, and the device name of the file system that is being backed up. If the storage medium (such as a tape) is too small to accommodate the backup, **dump** will pause and let you insert another. **dump** supports backups on multiple volumes. The **u** option will record the backup in the **/etc/dumpdates** file. In the following example, an entire backup (dump level 0) is performed on the file system on the **/dev/sda3** hard disk partition. The backup is stored on a tape device, **/dev/tape**.

```
dump -0u -f /dev/tape /dev/sda5
```

Note: You can use the **mt** command to control your tape device; **mt** has options to rewind, erase, and position the tape. The **rmt** command controls a remote tape device.

The storage device can be another hard disk partition, but it is usually a tape device. When you installed your system, your system most likely detected the device and set up **/dev/tape** as a link to it (just as it did with your CD-ROMs). If the link was not set up, you have to create it yourself or use the device name directly. Tape devices can have different device names, depending on the model or interface. SCSI tape devices are labeled with the prefix **st**, with a number attached for the particular device: **st0** is the first SCSI tape device. To use it in the **dump** command, just specify its name.

```
dump -0u -f /dev/st0 /dev/sda5
```

Should you need to back up to a device located on another system on your network, you would have to specify that hostname for the system and the name of its device. The hostname is entered before the device name and delimited with a colon. In the following example, the user backs up file system **/dev/sda4** to the SCSI tape device with the name **/dev/st0** on the **rabbit.mytrek.com** system:

```
dump -0u -f rabbit.mytrek.com:/dev/st0 /dev/sda4
```

The **dump** command works on one file system at a time. If your system has more than one file system, you will need to issue a separate **dump** command for each.

Tip: You can use the system **cron** utility to schedule backups using dump at specified times.

Operation	Description
-C	Lets you check a backup by comparing files on a file system with those in a backup.
-i	The interactive mode for restoring particular files and directories in a backup. A shell interface is generated where the user can use commands to specify file and directories to restore (see Table 18-5).
-R	Instructs **restore** to request a tape that is part of a multivolume backup, from which to continue the restore operation. Helpful when multivolume restore operations are interrupted.
-r	Restores a file system. Make sure that a newly formatted partition has been mounted and that you have changed to its top directory.
-t	Lists the contents of a backup or specified files in it.
-x	Extracts specified files or directories from a backup. A directory is restored with all its subdirectories. If no file or directory is specified, the entire file system is restored.
Additional Option	**Description**
-b *blocksize*	Use a specific block size; otherwise, restore will dynamically determine it from the block device.
-f *file/device*	Restores the backup on the specified file or device. Specify a hostname for remote devices.
-F *script*	Runs a script at the beginning of the restore.
-k	Uses Kerberos authentication for remote devices.
-h	Extracts only the specified directories, without their subdirectories.
-M *file/device*	Restores from multivolume backups, where the *file* is treated as a prefix and the suffix is a numbered sequence, *file*001, *file*002.
-N	Displays the names of files and directories, does not extract them.
-T *directory*	Specifies a directory to use for the storage of temporary files. The default value is **/tmp**.
-v	The verbose mode, where each file and its file type that **restore** operates on is displayed.
-y	By default, **restore** will query the operator to continue if an error occurs, such as bad blocks. This option suppresses that query, allowing **restore** to automatically continue.

Table 18-4: Operations and Options for restore

Recovering Backups

You use the **restore** command either to restore an entire file system or to just retrieve particular files. **restore** will extract files or directories from a backup archive and copy them to the current working directory. Make sure you are in the directory you want the files restored to

when you run **restore**. **restore** will also generate any subdirectories as needed. **restore** has several options for managing the restore operation (see Table 18-4).

To recover individual files and directories, you run **restore** in an interactive mode using the **-i** option. This will generate a shell with all the directories and files on the tape, letting you select the ones you want to restore. When you are finished, **restore** will then retrieve from a backup only those files you selected. This shell has its own set of commands that you can use to select and extract files and directories (see Table 18-5). The following command will generate an interactive interface listing all the directories and files backed up on the tape in the **/dev/tape** device:

```
restore -ivf /dev/tape
```

Command	Description
add [*arg*]	Adds files or directories to the list of files to be extracted. Such tagged files display an * before their names when listed with **ls**. All subdirectories of a tagged directory are also extracted.
cd *arg*	Changes the current working directory.
delete [*arg*]	Deletes a file or directory from the extraction list. All subdirectories for deleted directories will also be removed.
extract	Extracts files and directories on the extraction list.
help	Displays a list of available commands.
ls [*arg*]	Lists the contents of the current working directory or a specified directory.
pwd	Displays the full pathname of the current working directory.
quit	Exits the restore interactive mode shell. The **quit** command does not perform any extraction, even if the extraction list still has items in it.
setmodes	Sets the owner, modes, and times for all files and directories in the extraction list. Used to clean up an interrupted restore.
verbose	In the verbose mode, each file is listed as it is extracted. Also, the **ls** command lists the inode numbers for files and directories.

Table 18-5: Interactive Mode Shell Commands for restore

This command will generate a shell encompassing the entire directory structure of the backup. You are given a shell prompt and can use the **cd** command to move to different directories, and the **ls** command to list files and subdirectories. You use the **add** command to tag a file or directory for extraction. Should you later decide not to extract it, you can use the **delete** command to remove from the tagged list. Once you have selected all the items you want, you enter the **extract** command to retrieve them from the backup archive. To quit the restore shell, you enter **quit**. The **help** command will list the restore shell commands.

If you need to restore an entire file system, you would use restore with the **-r** option. You can restore the file system to any blank formatted hard disk partition of adequate size, including the

file system's original partition. If may be advisable, if possible, to restore the file system on another partition and check the results.

Restoring an entire file system involves setting up a formatted partition, mounting it to your system, and then changing to its top directory to run the **restore** command. First you should use **mkfs** to format the partition where you are restoring the file system, and then mount it onto your system. Then you use restore with the **-r** option and the **-f** option to specify the device holding the file system's backup. In the next example, the user formats and mounts the **/dev/sda5** partition and then restores on that partition the file system backup, currently on a tape in the **/dev/tape** device.

```
mkfs /dev/sda5
mount /dev/sda5 /mystuff
cd /mystuff
restore -rf /dev/tape
```

To restore from a backup device located on another system on your network, you would have to specify that hostname for the system and the name of its device. The hostname is entered before the device name and delimited with a colon. In the following example, the user restores a file system from the backup on the tape device with the name **/dev/tape** on the **rabbit.mytrek.com** system:

```
restore -rf rabbit.mytrek.com:/dev/tape
```

fedora

19. Shell Configuration

Four different major shells are commonly used on Linux systems: the Bourne Again shell (BASH), the AT&T Korn shell, the TCSH shell, and the Z shell. The BASH shell is an advanced version of the Bourne shell, which includes most of the advanced features developed for the Korn shell and the C shell. TCSH is an enhanced version of the C shell, originally developed for BSD versions of UNIX. The AT&T UNIX Korn shell is open source. The Z shell is an enhanced version of the Korn shell. Although their UNIX counterparts differ greatly, the Linux shells share many of the same features. In UNIX, the Bourne shell lacks many capabilities found in the other UNIX shells. In Linux, however, the BASH shell incorporates all the advanced features of the Korn shell and C shell, as well as the TCSH shell. All four shells are available for your use, though the BASH shell is the default.

Features	Description
bash	BASH shell, **/bin/bash**
bsh	BASH shell, **/bin/bsh** (link to **/bin/bash**)
sh	BASH shell, **/bin/sh** (link to **/bin/bash**)
tcsh	TCSH shell, **/usr/tcsh**
csh	TCSH shell , **/bin/csh** (link to **/bin/tcsh**)
ksh	Korn shell, **/bin/ksh** (also added link **/usr/bin/ksh**)
zsh	Z shell, **/bin/zsh**

Table 19-1: Shell Invocation Command Names

The BASH shell is the default shell for most Linux distributions. If you are logging in to a command line interface, you will be placed in the default shell automatically and given a shell prompt at which to enter your commands. The shell prompt for the BASH shell is a dollar sign (**$**). In the GUI interface, such as GNOME or KDE, you can open a terminal window that will display a command line interface with the prompt for the default shell (BASH). Though you log in to your default shell or display it automatically in a terminal window, you can change to another shell by entering its name. **tcsh** invokes the TCSH shell, **bash** the BASH shell, **ksh** the Korn shell, and **zsh** the Z shell. You can leave a shell by pressing CTRL-D or using the **exit** command. You only need one type of shell to do your work. Table 19-1 shows the different commands you can use to invoke different shells. Some shells have added links you can use to invoke the same shell, like **sh** and **bsh**, which link to and invoke the **bash** command for the BASH shell.

This chapter describes common features of the BASH shell, such as aliases, as well as how to configure the shell to your own needs using shell variables and initialization files. The other shells share many of the same features and use similar variables and initialization files.

Though the basic shell features and configurations are shown here, you should consult the respective online manuals and FAQs for each shell for more detailed examples and explanations.

Shell Initialization and Configuration Files

Each type of shell has its own set of initialization and configuration files. The TCSH shell uses **.login**, **.tcshrc**, and **.logout** files in place of **.profile**, **.bashrc**, and **.bash_logout**. The Z shell

has several initialization files: **.zshenv**, **.zlogin**, **.zprofile**, **.zschrc**, and **.zlogout**. See Table 19-2 for a listing. Check the Man pages for each shell to see how they are usually configured. When you install a shell, default versions of these files are automatically placed in the users' home directories. Except for the TCSH shell, all shells use much the same syntax for variable definitions and assigning values (TCSH uses a slightly different syntax, described in its Man pages).

Filename	Function
BASH Shell	
.bash_profile	Login initialization file
.bashrc	BASH shell configuration file
.bash_logout	Logout name
.bash_history	History file
/etc/profile	System login initialization file
/etc/bashrc	System BASH shell configuration file
/etc/profile.d	Directory for specialized BASH shell configuration files
/etc/bash_completion	Completion options for applications
TCSH Shell	
.login	Login initialization file
.tcshrc	TCSH shell configuration file
.logout	Logout file
Z Shell	
.zshenv	Shell login file (first read)
.zprofile	Login initialization file
.zlogin	Shell login file
.zshrc	Z shell configuration file
.zlogout	Logout file
Korn Shell	
.profile	Login initialization file
.kshrc	KORN shell configuration file

Table 19-2: Shell Configuration Files

Configuration Directories and Files

Applications often install configuration files in a user's home directory that contain specific configuration information, which tailors the application to the needs of that particular user. This may take the form of a single configuration file that begins with a period, or a directory that contains several configuration files. The directory name will also begin with a period. For example,

Mozilla installs a directory called **.mozilla** in the user's home directory that contains configuration files. On the other hand, many mail applications uses a single file called **.mailrc** to hold alias and feature settings set up by the user, though others like Evolution also have their own, **.evolution**. Most single configuration files end in the letters **rc**. **FTP** uses a file called **.netrc**. Most newsreaders use a file called **.newsrc**. Entries in configuration files are usually set by the application, though you can usually make entries directly by editing the file. Applications have their own set of special variables to which you can define and assign values. You can list the configuration files in your home directory with the **ls -a** command.

Aliases

You use the **alias** command to create another name for a command. The **alias** command operates like a macro that expands to the command it represents. The alias does not literally replace the name of the command; it simply gives another name to that command. An **alias** command begins with the keyword **alias** and the new name for the command, followed by an equal sign and the command the alias will reference.

Note: No spaces can be around the equal sign used in the **alias** command.

In the next example, **list** becomes another name for the **ls** command:

```
$ alias list=ls
$ ls
mydata today
$ list
mydata today
$
```

If you want an alias to be automatically defined, you have to enter the alias operation in a shell configuration file. On Fedora, aliases are defined in either the user's **.bashrc** file or in a **.bash_aliases** file. To use a **.bash_aliases** file, you have to first uncomment the commands in the .bashrc file that will read the **.bash_aliases** file. Just edit the **.bashrc** file and remove the preceding # so it appears like the following:

```
if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi
```

You can also place aliases in the **.bashrc** file directly. Some are already defined, though commented out. You can edit the **.bashrc** file and remove the # comment symbols from those lines to activate the aliases.

```
# some more ls aliases
alias ll='ls -l'
alias la='ls -A'
alias l='ls -CF'
```

Aliasing Commands and Options

You can also use an alias to substitute for a command and its option, but you need to enclose both the command and the option within single quotes. Any command you alias that contains spaces must be enclosed in single quotes as well. In the next example, the alias **lss**

references the **ls** command with its **-s** option, and the alias **lsa** references the **ls** command with the **-F** option. The **ls** command with the **-s** option lists files and their sizes in blocks, and **ls** with the **-F** option places a slash after directory names. Notice how single quotes enclose the command and its option.

```
$ alias lss='ls -s'
$ lss
mydata 14   today  6   reports  1
$ alias lsa='ls -F'
$ lsa
mydata today reports/
$
```

Aliases are helpful for simplifying complex operations. In the next example, **listlong** becomes another name for the **ls** command with the **-l** option (the long format that lists all file information), as well as the **-h** option for using a human-readable format for file sizes. Be sure to encase the command and its arguments within single quotes so that they are taken as one argument and not parsed by the shell.

```
$ alias listlong='ls -lh'
$ listlong
-rw-r--r--    1 root   root   51K  Sep  18  2008 mydata
-rw-r--r--    1 root   root   16K  Sep  27  2008 today
```

Aliasing Commands and Arguments

You may often use an alias to include a command name with an argument. If you execute a command that has an argument with a complex combination of special characters on a regular basis, you may want to alias it. For example, suppose you often list just your source code and object code files—those files ending in either a **.c** or **.o**. You would need to use as an argument for **ls** a combination of special characters such as ***.[co]**. Instead, you can alias **ls** with the **.[co]** argument, giving it a simple name. In the next example, the user creates an alias called **lsc** for the command **ls.[co]**:

```
$ alias lsc='ls *.[co]'
$ lsc
main.c main.o lib.c lib.o
```

Aliasing Commands

You can also use the name of a command as an alias. This can be helpful in cases where you should use a command only with a specific option. In the case of the **rm**, **cp**, and **mv** commands, the **-i** option should always be used to ensure an existing file is not overwritten. Instead of always being careful to use the **-i** option each time you use one of these commands, you can alias the command name to include the option. In the next example, the **rm**, **cp**, and **mv** commands have been aliased to include the **-i** option:

```
$ alias rm='rm -i'
$ alias mv='mv -i'
$ alias cp='cp -i'
```

The **alias** command by itself provides a list of all aliases that have been defined, showing the commands they represent. You can remove an alias by using the **unalias** command. In the next example, the user lists the current aliases and then removes the **lsa** alias:

```
$ alias
lsa=ls -F
list=ls
rm=rm -i
$ unalias lsa
```

Controlling Shell Operations

The BASH shell has several features that enable you to control the way different shell operations work. For example, setting the **noclobber** feature prevents redirection from overwriting files. You can turn these features on and off like a toggle, using the **set** command. The **set** command takes two arguments: an option specifying on or off and the name of the feature. To set a feature on, you use the **-o** option, and to set it off, you use the **+o** option. Here is the basic form:

```
$ set -o feature        turn the feature on
$ set +o feature        turn the feature off
```

Features	Description
$ set -/+o *feature*	BASH shell features are turned on and off with the **set** command; **-o** sets a feature on and **+o** turns it off: **$ set -o noclobber** *set noclobber on* **$ set +o noclobber** *set noclobber off*
ignoreeof	Disables CTRL-D logout
noclobber	Does not overwrite files through redirection
noglob	Disables special characters used for filename expansion: *****, **?**, **~**, and **[]**

Table 19-3: BASH Shell Special Features

Three of the most common features are **ignoreeof**, **noclobber**, and **noglob**. Table 19-3 lists these different features, as well as the **set** command. Setting **ignoreeof** enables a feature that prevents you from logging out of the user shell with CTRL-D. CTRL-D is not only used to log out of the user shell, but also to end user input entered directly into the standard input. CTRL-D is used often for the Mail program or for utilities such as **cat**. You can easily enter an extra CTRL-D in such circumstances and accidentally log yourself out. The **ignoreeof** feature prevents such accidental logouts. In the next example, the **ignoreeof** feature is turned on using the **set** command with the **-o** option. The user can then log out only by entering the **logout** command.

```
$ set -o ignoreeof
$ CTRL-D
Use exit to logout
$
```

Environment Variables and Subshells: export

When you log in to your account, Linux generates your user shell. Within this shell, you can issue commands and declare variables. You can also create and execute shell scripts. When you execute a shell script, however, the system generates a subshell. You then have two shells, the one you logged in to and the one generated for the script. Within the script shell, you can execute another shell script, which then has its own shell. When a script has finished execution, its shell terminates and you return to the shell from which it was executed. In this sense, you can have many shells, each nested within the other. Variables you define within a shell are local to it. If you define a variable in a shell script, then, when the script is run, the variable is defined with that script's shell and is local to it. No other shell can reference that variable. In a sense, the variable is hidden within its shell.

You can define environment variables in all types of shells, including the BASH shell, the Z shell, and the TCSH shell. The strategy used to implement environment variables in the BASH shell, however, is different from that of the TCSH shell. In the BASH shell, environment variables are exported. That is to say, a copy of an environment variable is made in each subshell. For example, if the **EDITOR** variable is exported, a copy is automatically defined in each subshell for you. In the TCSH shell, on the other hand, an environment variable is defined only once and can be directly referenced by any subshell.

Shell Variables	Description
BASH	Holds full pathname of BASH command
BASH_VERSION	Displays the current BASH version number
GROUPS	Groups that the user belongs to
HISTCMD	Number of the current command in the history list
HOME	Pathname for user's home directory
HOSTNAME	The hostname
HOSTTYPE	Displays the type of machine the host runs on
OLDPWD	Previous working directory
OSTYPE	Operating system in use
PATH	List of pathnames for directories searched for executable commands
PPID	Process ID for shell's parent shell
PWD	User's working directory
RANDOM	Generates random number when referenced
SHLVL	Current shell level, number of shells invoked
UID	User ID of the current user

Table 19-4: Shell Variables, Set by the Shell

In the BASH shell, an environment variable can be thought of as a regular variable with added capabilities. To make an environment variable, you apply the **export** command to a variable

you have already defined. The **export** command instructs the system to define a copy of that variable for each new shell generated. Each new shell will have its own copy of the environment variable. This process is called *exporting variables*. To think of exported environment variables as global variables is a mistake. A new shell can never reference a variable outside of itself. Instead, a copy of the variable with its value is generated for the new shell.

Configuring Your Shell with Shell Parameters

When you log in, Linux will set certain parameters for your login shell. These parameters can take the form of variables or features. See the previous section "Controlling Shell Operations" for a description of how to set features. Linux reserves a predefined set of variables for shell and system use. These are assigned system values, in effect, setting parameters. Linux sets up parameter shell variables you can use to configure your user shell. Many of these parameter shell variables are defined by the system when you log in. Some parameter shell variables are set by the shell automatically, and others are set by initialization scripts, described later. Certain shell variables are set directly by the shell, and others are simply used by it. Many of these other variables are application specific, used for such tasks as mail, history, or editing. Functionally, it may be better to think of these as system-level variables, as they are used to configure your entire system, setting values such as the location of executable commands on your system, or the number of history commands allowable. See Table 19-4 for a list of those shell variables set by the shell for shell-specific tasks; Table 19-5 lists those used by the shell for supporting other applications.

Shell Variables	Description
BASH_VERSION	Displays the current BASH version number
CDPATH	Search path for the **cd** command
EXINIT	Initialization commands for Ex/Vi editor
FCEDIT	Editor used by the history **fc** command.
GROUPS	Groups that the user belongs to
HISTFILE	The pathname of the history file
HISTSIZE	Number of commands allowed for history
HISTFILESIZE	Size of the history file in lines
HISTCMD	Number of the current command in the history list
HOME	Pathname for user's home directory
HOSTFILE	Sets the name of the hosts file, if other than **/etc/hosts**
IFS	Interfield delimiter symbol
IGNOREEOF	If not set, EOF character will close the shell. Can be set to the number of EOF characters to ignore before accepting one to close the shell (default is 10)
INPUTRC	Set the **inputrc** configuration file for Readline (command line). Default is current directory, **.inputrc**. Most Linux distributions set this to **/etc/inputrc**
KDEDIR	The pathname location for the KDE desktop
LOGNAME	Login name

MAIL	Name of specific mail file checked by Mail utility for received messages, if MAILPATH is not set
MAILCHECK	Interval for checking for received mail
MAILPATH	List of mail files to be checked by Mail for received messages
HOSTTYPE	Linux platforms, such as i686, x86_64, or ppc
PROMPT_COMMAND	Command to be executed before each prompt, integrating the result as part of the prompt
HISTFILE	The pathname of the history file
PS1	Primary shell prompt
PS2	Secondary shell prompt
QTDIR	Location of the Qt library (used for KDE)
SHELL	Pathname of program for type of shell you are using
TERM	Terminal type
TMOUT	Time that the shell remains active awaiting input
USER	Username
UID	Real user ID (numeric)
EUID	Effective user ID (numeric). This is usually the same as the UID but can be different when the user changes IDs, as with the **su** command, which allows a user to become an effective root user

Table 19-5: System Environment Variables Used by the Shell

A reserved set of keywords is used for the names of these system variables. You should not use these keywords as the names of any of your own variable names. The system shell variables are all specified in uppercase letters, making them easy to identify. Shell feature variables are in lowercase letters. For example, the keyword **HOME** is used by the system to define the **HOME** variable. **HOME** is a special environment variable that holds the pathname of the user's home directory. On the other hand, the keyword **noclobber** is used to set the **noclobber** feature on or off.

Shell Parameter Variables

Many of the shell parameter variables automatically defined and assigned initial values by the system when you log in can be changed, if you wish. Some parameter variables exist whose values should not be changed, however. For example, the **HOME** variable holds the pathname for your home directory. Commands such as **cd** reference the pathname in the **HOME** shell variable to locate your home directory. Some of the more common of these parameter variables are described in this section. Other parameter variables are defined by the system and given an initial value that you are free to change. To do this, you redefine them and assign a new value. For example, the **PATH** variable is defined by the system and given an initial value; it contains the pathnames of directories where commands are located. Whenever you execute a command, the shell searches for it in these directories. You can add a new directory to be searched by redefining the **PATH** variable yourself, so that it will include the new directory's pathname. Still other parameter variables exist that the system does not define. These are usually optional features, such as the **EXINIT** variable

that enables you to set options for the Vi editor. Each time you log in, you must define and assign a value to such variables. Some of the more common parameter variables are **SHELL**, **PATH**, **PS1**, **PS2**, and **MAIL**. The **SHELL** variable holds the pathname of the program for the type of shell you log in to. The **PATH** variable lists the different directories to be searched for a Linux command. The **PS1** and **PS2** variables hold the prompt symbols. The **MAIL** variable holds the pathname of your mailbox file. You can modify the values for any of them to customize your shell.

Note: You can obtain a listing of the currently defined shell variables using the **env** command. The **env** command operates like the **set** command, but it lists only parameter variables.

Using Initialization Files

You can automatically define parameter variables using special shell scripts called initialization files. An *initialization file* is a specially named shell script executed whenever you enter a certain shell. You can edit the initialization file and place in it definitions and assignments for parameter variables. When you enter the shell, the initialization file will execute these definitions and assignments, effectively initializing parameter variables with your own values. For example, the BASH shell's **.bash_profile** file is an initialization file executed every time you log in. It contains definitions and assignments of parameter variables. However, the **.bash_profile** file is basically only a shell script, which you can edit with any text editor such as the Vi editor; changing, if you wish, the values assigned to parameter variables.

In the BASH shell, all the parameter variables are designed to be environment variables. When you define or redefine a parameter variable, you also need to export it to make it an environment variable. This means any change you make to a parameter variable must be accompanied by an **export** command. You will see that at the end of the login initialization file, **.bash_profile**, there is usually an **export** command for all the parameter variables defined in it.

Your Home Directory: HOME

The **HOME** variable contains the pathname of your home directory. Your home directory is determined by the parameter administrator when your account is created. The pathname for your home directory is automatically read into your **HOME** variable when you log in. In the next example, the **echo** command displays the contents of the **HOME** variable:

```
$ echo $HOME
/home/chris
```

The **HOME** variable is often used when you need to specify the absolute pathname of your home directory. In the next example, the absolute pathname of **reports** is specified using **HOME** for the home directory's path:

```
$ ls $HOME/reports
```

Command Locations: PATH

The **PATH** variable contains a series of directory paths separated by colons. Each time a command is executed, the paths listed in the **PATH** variable are searched one by one for that command. For example, the **cp** command resides on the system in the directory **/bin**. This directory path is one of the directories listed in the **PATH** variable. Each time you execute the **cp** command,

this path is searched and the **cp** command located. The system defines and assigns **PATH** an initial set of pathnames. In Linux, the initial pathnames are **/bin** and **/usr/bin**.

The shell can execute any executable file, including programs and scripts you have created. For this reason, the **PATH** variable can also reference your working directory; so if you want to execute one of your own scripts or programs in your working directory, the shell can locate it. No spaces are allowed between the pathnames in the string. A colon with no pathname specified references your working directory. Usually, a single colon is placed at the end of the pathnames as an empty entry specifying your working directory. For example, the pathname **//bin:/usr/bin:** references three directories: **/bin**, **/usr/bin**, and your current working directory.

```
$ echo $PATH
/bin:/usr/sbin:
```

You can add any new directory path you want to the **PATH** variable. This can be useful if you have created several of your own Linux commands using shell scripts. You can place these new shell script commands in a directory you create and then add that directory to the **PATH** list. Then, no matter what directory you are in, you can execute one of your shell scripts. The **PATH** variable will contain the directory for that script, so that directory will be searched each time you issue a command.

You add a directory to the **PATH** variable with a variable assignment. You can execute this assignment directly in your shell. In the next example, the user **chris** adds a new directory, called **bin,** to the **PATH**. Although you could carefully type in the complete pathnames listed in **PATH** for the assignment, you can also use an evaluation of **PATH**—**$PATH**—in its place. In this example, an evaluation of **HOME** is also used to designate the user's home directory in the new directory's pathname. Notice the last colon, which specifies the working directory:

```
$ PATH=$PATH:$HOME/mybin:
$ export PATH
$ echo $PATH
/bin:/usr/bin::/home/chris/mybin
```

If you add a directory to **PATH** yourself while you are logged in, the directory will be added only for the duration of your login session. When you log back in, the login initialization file, **.bash_profile**, will again initialize your **PATH** with its original set of directories. The **.bash_profile** file is described in detail a bit later in this chapter. To add a new directory to your **PATH** permanently, you need to edit your **.bash_profile** file and find the assignment for the **PATH** variable. Then, you simply insert the directory, preceded by a colon, into the set of pathnames assigned to **PATH**.

Specifying the BASH Environment: BASH_ENV

The **BASH_ENV** variable holds the name of the BASH shell initialization file to be executed whenever a BASH shell is generated. For example, when a BASH shell script is executed, the **BASH_ENV** variable is checked and the name of the script that it holds is executed before the shell script. The **BASH_ENV** variable usually holds **$HOME/.bashrc**. This is the **.bashrc** file in the user's home directory. (The **.bashrc** file is discussed later in this chapter.) You can specify a different file if you wish, using that instead of the **.bashrc** file for BASH shell scripts.

Configuring the Shell Prompt

The **PS1** and **PS2** variables contain the primary and secondary prompt symbols, respectively. The primary prompt symbol for the BASH shell is a dollar sign (**$**). You can change the prompt symbol by assigning a new set of characters to the **PS1** variable. In the next example, the shell prompt is changed to the **->** symbol:

```
$ PS1= '->'
-> export PS1
->
```

The following table lists the codes for configuring your prompt:

Prompt Codes	Description
\!	Current history number
\$	Use **$** as prompt for all users except the root user, which has the **#** as its prompt
\d	Current date
\#	History command number for just the current shell
\h	Hostname
\s	Shell type currently active
\t	Time of day in hours, minutes, and seconds.
\u	Username
\v	Shell version
\w	Full pathname of the current working directory
\W	Name of the current working directory
****	Displays a backslash character
\n	Inserts a newline
\[\]	Allows entry of terminal specific display characters for features like color or bold font
*****nnn*	Character specified in octal format

You can change the prompt to be any set of characters, including a string, as shown in the next example:

```
$ PS1="Please enter a command: "
Please enter a command: export PS1
Please enter a command: ls
mydata /reports
Please enter a command:
```

The PS2 variable holds the secondary prompt symbol, which is used for commands that take several lines to complete. The default secondary prompt is >. The added command lines begin with the secondary prompt instead of the primary prompt. You can change the secondary prompt just as easily as the primary prompt, as shown here:

```
$ PS2="@"
```

Like the TCSH shell, the BASH shell provides you with a predefined set of codes you can use to configure your prompt. With them you can make the time, your username, or your directory pathname a part of your prompt. You can even have your prompt display the history event number of the current command you are about to enter. Each code is preceded by a \ symbol: **\w** represents the current working directory, **\t** the time, and **\u** your username; **\!** will display the next history event number. In the next example, the user adds the current working directory to the prompt:

```
$ PS1="\w $"
/home/dylan $
```

The codes must be included within a quoted string. If no quotes exist, the code characters are not evaluated and are themselves used as the prompt. **PS1=\w** sets the prompt to the characters **\w**, not the working directory. The next example incorporates both the time and the history event number with a new prompt:

```
$ PS1="\t \! ->"
```

The default BASH prompt is **\s-\v\$** to display the type of shell, the shell version, and the $ symbol as the prompt. Some distributions, including Fedora, have changed this to a more complex command consisting of the user, the hostname, and the name of the current working directory. A sample configuration is shown here. A simple equivalent is show here with @ sign in the hostname and a $ for the final prompt symbol. The home directory is represented with a tilde (**~**).

```
$ PS1="\u@\h:\w$"
richard@turtle.com:~$
```

Fedora also includes some complex prompt definitions in the **.bashrc** file to support color prompts and detect any remote user logins.

Specifying Your News Server

Several shell parameter variables are used to set values used by network applications, such as web browsers or newsreaders. **NNTPSERVER** is used to set the value of a remote news server accessible on your network. If you are using an ISP, the ISP usually provides a Usenet news server you can access with your newsreader applications. However, you first have to provide your newsreaders with the Internet address of the news server. This is the role of the **NNTPSERVER** variable. News servers on the Internet usually use the NNTP protocol. **NNTPSERVER** should hold the address of such a news server. For many ISPs, the news server address is a domain name that begins with **nntp**. The following example assigns the news server address **nntp.myservice.com** to the **NNTPSERVER** shell variable. Newsreader applications automatically obtain the news server address from **NNTPSERVER**. Usually, this assignment is placed in the shell initialization file, **.bash_profile**, so that it is automatically set each time a user logs in.

```
NNTPSERVER=news.myservice.com
export NNTPSERVER
```

Configuring Your Login Shell: .bash_profile

The **.bash_profile** file is the BASH shell's login initialization file. It is a script file that is automatically executed whenever a user logs in. The file contains shell commands that define

system environment variables used to manage your shell. They may be either redefinitions of system-defined variables or definitions of user-defined variables. For example, when you log in, your user shell needs to know what directories hold Linux commands. It will reference the **PATH** variable to find the pathnames for these directories. However, first, the **PATH** variable must be assigned those pathnames. In the **.bash_profile** file, an assignment operation does just this. Because it is in the **.bash_profile** file, the assignment is executed automatically when the user logs in. The root user .bash_profile script adds an operation to undefine the USERNAME variable for security purposes, **unset USERNAME**.

Exporting Variables

Any new parameter variables you may add to the **.bash_profile** file, will also need to be exported, using the **export** command. This makes them accessible to any subshells you may enter. You can export several variables in one **export** command by listing them as arguments. The **.bash_profile** file contain no variable definitions, though you can add ones of your own. In this case, the **.bash_profile** file would have an **export** command with a list of all the variables defined in the file. If a variable is missing from this list, you may be unable to access it. The **.bashrc** file contain a definition of the **HISTCONTROL** variable, which is then exported. You can also combine the assignment and **export** command into one operation as shown here for **NNTPSERVER**:

```
export NNTPSERVER=news.myservice.com
```

Variable Assignments

A copy of the standard **.bash_profile** file, provided for you when your account is created, is listed in the next example. Notice how **PATH** is assigned. **PATH** is a parameter variable the system has already defined. **PATH** holds the pathnames of directories searched for any command you enter. The assignment **PATH=$PATH:$HOME/bin** has the effect of redefining **PATH** to include your **bin** directory within your home directory so that your **bin** directory will also be searched for any commands, including ones you create yourself, such as scripts or programs.

.bash_profile

```
# .bash_profile

# Get the aliases and functions
if [  -f  ~/.bashrc ]; then
          . ~/.bashrc
fi

# User specific environment and startup programs
PATH=$PATH:$HOME/bin
export PATH
```

Should you want to have your current working directory searched also, you can use any text editor to modify this line in your **.bash_profile** file **PATH=$PATH:$HOME/bin**. You would insert a colon **:** after **bin** In fact, you can change this entry to add as many directories as you want to search. Making commands automatically executable in your current working directory could be a security risk, allowing files in any directory to be executed, instead of in certain specified

directories. An example of how to modify your **.bash_profile** file is shown in the following section.

```
PATH=$PATH;$HOME/bin:
```

Editing Your BASH Profile Script

Your **.bash_profile** initialization file is a text file that can be edited by a text editor, like any other text file. You can easily add new directories to your **PATH** by editing **.bash_profile** and using editing commands to insert a new directory pathname in the list of directory pathnames assigned to the **PATH** variable. You can even add new variable definitions. If you do so, however, be sure to include the new variable's name in the **export** command's argument list. For example, if your **.bash_profile** file does not have any definition of the **EXINIT** variable, you can edit the file and add a new line that assigns a value to **EXINIT**. The definition **EXINIT='set nu ai'** will configure the Vi editor with line numbering and indentation. You then need to add **EXINIT** to the **export** command's argument list. When the **.bash_profile** file executes again, the **EXINIT** variable will be set to the command **set nu ai**. When the Vi editor is invoked, the command in the **EXINIT** variable will be executed, setting the line number and auto-indent options automatically.

In the following example, the user's **.bash_profile** has been modified to include definitions of **EXINIT** and redefinitions of **PATH**, **PS1**, and **HISTSIZE**. The **PATH** variable has the ending colon added to it that specifies the current working directory, enabling you to execute commands that may be located in either the home directory or the working directory. The redefinition of **HISTSIZE** reduces the number of history events saved, from 1,000 defined in the system's **.bash_profile** file, to 30. The redefinition of the **PS1** parameter variable changes the prompt to just show the pathname of the current working directory. Any changes you make to parameter variables within your **.bash_profile** file override those made earlier by the system's **.bash_profile** file. All these parameter variables are then exported with the **export** command.

.bash_profile

```
# .bash_profile

# Get the aliases and functions
if [  -f  ~/.bashrc ]; then
          . ~/.bashrc
fi
PATH=$PATH:$HOME/bin:

HISTSIZE=30
NNTPSERVER=news.myserver.com
EXINIT='set nu ai'
PS1="\w \$"
export PATH HISTSIZE EXINIT PS1 NNTPSERVER
```

Manually Re-executing the .bash_profile Script

Although the **.bash_profile** file is executed each time you log in, it is not automatically re-executed after you make changes to it. The **.bash_profile** file is an initialization file that is executed *only* whenever you log in. If you want to take advantage of any changes you make to it without having to log out and log in again, you can re-execute the **.bash_profile** file with the dot

(**.**) command. The **.bash_profile** file is a shell script and, like any shell script, can be executed with the **.** command.

```
$ . .bash_profile
```

Alternatively, you can use the **source** command to execute the **.bash_profile** initialization file or any initialization file such as **.login** used in the TCSH shell or **.bash_bashrc**.

```
$ source .bash_profile
```

System Shell Profile Script

Your Linux system also has its own profile file that it executes whenever any user logs in. This system initialization file is simply called **profile** and is found in the **/etc** directory, **/etc/profile**. This file contains parameter variable definitions the system needs to provide for each user. A copy of the system's **.profile** file follows. Fedora and Red Hat use a **pathmunge** function to generate a directory list for the **PATH** variable. Normal user paths will lack the system directories (those with **sbin** in the path) but include the name of their home directory, along with **/usr/kerberos/bin** for Kerberos tools. The path generated for the root user (Effective User ID of 0, EUID = 0) will include both system and user application directories, adding **/usr/kerberos/sbin**, **/sbin**, **/usr/sbin**, and **/usr/local/sbin**, as well as the root user local application directory, **/root/bin**.

```
# echo $PATH
/usr/kerberos/bin/usr/local/bin:usr/sbin:/bin:/usr/X11R6/bin:/home/richard/bin
```

A special work-around is included for the Korn Shell to set the User and Effective User IDs (**EUID** and **UID**).

The **USER**, **MAIL**, and **LOGNAME** variables are then set, provided that **/usr/bin/id**, which provides the User ID, is executable. The **id** command with the **-un** option provides the user ID's text name only, like **chris** or **richard**.

HISTSIZE is also redefined to include a larger number of history events. An entry has been added here for the **NNTPSERVER** variable. Normally, a news server address is a value that needs to be set for all users. Such assignments should be made in the system's **/etc/profile** file by the system administrator, rather than in each individual user's own **.bash_profile** file.

Note: The **/etc/profile** file also executes any scripts in the directory **/etc/profile.d**. This design allows for a more modular structure. Rather than make entries by editing the **/etc/profile** file, you can just add a script to **profile.d** directory.

The **/etc/profile** file also runs the **/etc/inputrc** file, which configures your command line editor. Here you will find key assignments for different tasks, such as moving to the end of a line or deleting characters. Global options are set as well. Keys are represented in hexadecimal format.

/etc/profile

```
# /etc/profile

# System-wide environment and startup programs, for login setup
# Functions and aliases go in /etc/bashrc
```

```
pathmunge () {
          if ! echo $PATH | /bin/egrep -q "(^|:)$1($|:)" ; then
              if [ "$2" = "after" ] ; then
                    PATH=$PATH:$1
             else
                    PATH=$1:$PATH
              fi
        fi
}

# ksh workaround
if [ -z "$EUID" -a -x /usr/bin/id ]; then
          EUID=`id -u`
          UID=`id -ru`
fi

# Path manipulation
if [ "$EUID" = "0" ]; then
          pathmunge /sbin
          pathmunge /usr/sbin
          pathmunge /usr/local/sbin
fi

# No core files by default
ulimit -S -c 0 > /dev/null 2>&1

if [ -x /usr/bin/id ]; then
          USER="`id -un`"
          LOGNAME=$USER
          MAIL="/var/spool/mail/$USER"
fi

HOSTNAME=`/bin/hostname`
HISTSIZE=1000

if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
    INPUTRC=/etc/inputrc
fi

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC

for i in /etc/profile.d/*.sh ; do
    if [ -r "$i" ]; then
        if [ "$PS1" ]; then
          . $i
        else
         . $i &>/dev/null
        fi
    fi
done

unset i
unset pathmunge
```

The number of aliases and variable settings needed for different applications would make the **/etc/profile** file much too large to manage. Instead, application task-specific aliases and variables are placed in separate configuration files located in the **/etc/profile.d** directory. There are corresponding scripts for both the BASH and C shells. The BASH shell scripts are run by **/etc/profile**. The scripts are named for the kinds of tasks and applications they configure. For example, **colorls.sh** sets the file type color coding when the **ls** command displays files and directories. The **vim.sh** file sets an alias for the **vi** command, executing **vim** whenever the user enters just **vi**. The **kde.sh** file sets the global environment variable **KDEDIR**, specifying the KDE Desktop applications directory, in this case **/usr**. The **krb5.sh** file adds the pathnames for Kerberos, **/usr/kerberos**, to the **PATH** variable. Files run by the BASH shell end in the extension **.sh**, and those run by the C shell have the extension **.csh**. The **/etc/profile** script will also check first if the **PS1** variable is defined before running any **/etc/profile.d** scripts.

Configuring the BASH Shell: .bashrc

The **.bashrc** file is a configuration file executed each time you enter the BASH shell or generate any subshells. If the BASH shell is your login shell, **.bashrc** is executed along with your **.bash_login** file when you log in. If you enter the BASH shell from another shell, the **.bashrc** file is automatically executed, and the variable and alias definitions it contains will be defined. If you enter a different type of shell, the configuration file for that shell will be executed instead. For example, if you were to enter the TCSH shell with the **tcsh** command, the **.tcshrc** configuration file would be executed instead of **.bashrc**.

The User .bashrc BASH Script

The **.bashrc** shell configuration file is actually executed each time you generate a BASH shell, such as when you run a shell script. In other words, each time a subshell is created, the **.bashrc** file is executed. This has the effect of exporting any local variables or aliases you have defined in the **.bashrc** shell initialization file. The **.bashrc** file usually contains the definition of aliases and any feature variables used to turn on shell features. Aliases and feature variables are locally defined within the shell. But the **.bashrc** file defines them in every shell. For this reason, the **.bashrc** file usually holds aliases and options you want defined for each shell. As an example of how you can add your own aliases and options, aliases for the **rm**, **cp**, and **mv** commands and the shell **noclobber** and **ignoreeof** options have been added. For the root user **.bashrc**, the **rm**, **cp**, and **mv** aliases have already been included in the root's **.bashrc** file.

.bashrc

```
# .bashrc
# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
```

You can add any commands or definitions of your own to your **.bashrc** file. If you have made changes to **.bashrc** and you want them to take effect during your current login session, you need to re-execute the file with either the **.** or the **source** command.

```
$ . .bashrc
```

The System /etc/bashrc BASH Script

Fedora also has a system **bashrc** file executed for all users, called **/etc/bashrc**. Currently the **/etc/bashrc** file sets the default shell prompt, as well as instructions for checking whether a user is authorized to use a command. If in an interactive shell, **/etc/bashrc** file will determine the default prompt. If not in a login shell, it will then determine the PATH and run any profile.d scripts.

Note: The bash_completion files in the **/etc/bash_completion.d** directory provide command completion options for applications.

The BASH Shell Logout File: .bash_logout

The **.bash_logout** file is also a configuration file, but it is executed when the user logs out. It is designed to perform any operations you want to occur whenever you log out. Instead of variable definitions, the **.bash_logout** file usually contains shell commands that form a kind of shutdown procedure—actions you always want taken before you log out. One common logout command is to clear the screen and then issue a farewell message. This is performed by the root user **.bash_logout** file, though not in the normal user **.bash_logout** files.

.bash_logout

```
# ~/.bash_logout:

clear
echo "Good-bye for now"
```

As with **.bash_profile**, you can add your own shell commands to **.bash_logout**. The root user **.bash_logout** file includes instructions to invoke the **clear** command to clear the screen. In the next example, the user has added an **echo** command in the **.bash_logout** file.

Part 4: Network Services

fedora

20. Mail Servers

Mail Transport Agents

Postfix

Sendmail

POP and IMAP Server: Dovecot

Spam: SpamAssassin

Mail servers provide Internet users with electronic mail services. They have their own TCP/IP protocols such as the Simple Mail Transfer Protocol (SMTP), the Post Office Protocol (POP), and the Internet Mail Access Protocol (IMAP). Messages are sent across the Internet through mail servers that service local domains. A *domain* can be seen as a subnet of the larger Internet, with its own server to handle mail messages sent from or received for users on that subnet. When a user mails a message, it is first sent from his or her host system to the mail server. The mail server then sends the message to another mail server on the Internet, the one servicing the subnet on which the recipient user is located. The receiving mail server then sends the message to the recipient's host system.

At each stage, a different type of operation takes place using different agents (programs). A mail user agent (MUA) is a mail client program, such as Evolution, Thunderbird, Kmail, or mail. With an MUA, a user composes a mail message and sends it. Then a mail transfer agent (MTA) transports the messages over the Internet. MTAs are mail servers that use SMTP to send messages across the Internet from one mail server to another, transporting them among subnets. On Fedora, the commonly used MTAs are Postfix and Sendmail. These are mail server daemons that constantly checks for incoming messages from other mail servers and sends outgoing messages to appropriate servers (see Table 20-1). Incoming messages received by a mail server are distributed to a user with mail delivery agents (MDAs). Most Linux systems use procmail as their MDA, taking messages received by the mail server and delivering them to user accounts (see **www.procmail.org** for more information).

Mail Transport Agents

On Fedora you can install and configure either Postfix, Exim, or Sendmail. You can also set up your Linux system to run a POP server. POP servers hold users' mail until they log in to access their messages, instead of having mail sent to their hosts directly.

Agent	Description
Sendmail	Sendmail mail transfer agent, supported by the Sendmail consortium **www.sendmail.org**
Postfix	Fast, easy-to-configure, and secure mail transfer agent compatible with Sendmail and designed to replace it **www.postfix.org**
Qmail	Fast, flexible, and secure MTA with its own implementation and competitive with Postfix **www.qmail.org**
Exim	MTA based on smail3 **www.exim.org**
Courier	Courier MTA **www.courier-major.org**

Table 20-1: Mail Transfer Agents

Courier is a fast, small, and secure MTA that maintains some compatibility with Sendmail. The Courier software package also includes POP, IMAP, and webmail servers along

with mailing list services. It supports extensive authentication methods including shadow passwords, PAM, and LDAP.

Exim is a fast and flexible MTA similar to Sendmail. Developed at the University of Cambridge, it has a very different implementation than Sendmail.

Qmail is also a fast and secure MTA, but it has little compatibility with Sendmail. It has its own configuration and maintenance files. Like Postfix, it has a modular design, using a different program for each mail task. It also focuses on security, speed, and easy configuration.

Postfix

Postfix is a fast, secure, and flexible MTA designed to replace Sendmail while maintaining as much compatibility as possible. Written by Wietse Venema and originally released as the IBM Secure Mailer, it is now available under the GNU license (www.postfix.org). Postfix was created with security in mind, treating all incoming mail as potential security risks. Postfix uses many of the same Sendmail directories and files and makes use of Sendmail wrappers, letting Sendmail clients interact seamlessly with Postfix servers. Postfix is also easier than Sendmail to configure, using its own configuration file.

Instead of one large program, Postfix is implemented as a collection of smaller programs, each designed to perform a specific mail-related task. A Postfix master daemon runs continuously and manages the use of the other Postfix daemons, running them only as needed. A **bounce** daemon handles undeliverable mail, a **trivial-rewrite** daemon redirects messages, and the **showq** daemon provides information on the print queues.

Postfix Commands

Several Postfix commands allow you to manage your server tasks. The **sendmail** command sends messages. You use **mailq** to display the status of your mail queues. The **newaliases** command takes mail aliases listed in the aliases files and stores them in a database file that can be used by Postfix.

The **postmap** command is used to maintain various database files used by Postfix, such as the alias file for mail aliases and the access file that restricts messages received by the server. You can also implement these database files as SQL databases like MySQL, allowing for easier management. The **mysql_table** Man page provides detailed information on how to configure SQL database support (check **pgsql_table** for Postgresql database support). You could also use LDAP instead of SQL (**ldap_table**).

In addition, Postfix provides lower-level tools, all beginning with the term **post**, such as the **postalias** command, which maintains the alias database, and **postcat**, which displays print queue files.

Postfix Configuration: main.cf

Postfix configuration is handled by setting parameters in its configuration file, **main.cf**. In addition a **master.cf** file holds parameters for running Postfix services, and **dynamicmaps.cf** file for additional runtime capabilities.

A default **/etc/postfix/main.cf** file is installed with Postfix, with most of the essential configuration values already set. Parameter names tend to be user friendly. For example, directory locations are specified by parameters ending in the term **directory**, such as **queue_directory** for the location of Postfix queues and **daemon_directory** for the location of the Postfix daemons. Defaults are already implemented for most parameters. For example, defaults are set for particular resource controls, such as message size, time limits, and the number of allowed messages per queue. You can edit the **main.cf** file to change the parameter values to meet your own needs. After making any changes, you need only to reload the configuration using the **postfix reload** command:

```
postfix reload
```

Network Parameters

You will most likely need to set several network parameters. To ease this process, Postfix defines parameters that hold key network information, such as **myhostname**, which holds the hostname of your system, and **mydomain**, which holds the domain name of your network. For example, **myhostname** would be set to the host **turtle.mytrek.com**, whereas **mydomain** would be just **mytrek.com**. Parameters like **myhostname** and **mydomain** are themselves used as values assigned to other parameters. On Fedora, myhostname will be set to the system mail name you entered in the second configuration screen during the Postfix installation. In the next example, **myhostname** and **mydomain** are set to the host the mail server is running on and its network domain:

```
myhostname=turtle.mytrek.com
mydomain=mytrek.com
```

The **myorigin** parameter specifies the origin address for e-mail sent by the server. On Fedora this is commented your. You could set to the value of the parameter **myhostname**, as shown here. Note that a **$** precedes the **myhostname** variable to evaluate it.

```
myorigin=$myhostname
```

If you are using a single system directly attached to the Internet, you may want to keep this configuration, labeling mail as being sent by your host. However, if your system is operating as a gateway for a network, your mail server is sending out mail from different hosts on that network. You may wish to change the origin address to the domain name, so that mail is perceived as sent from the domain.

```
myorigin=$mydomain
```

Local Networks

The **mydestination** parameter holds the list of domains that your mail server will receive mail for. By default, these include **localhost** and your system's hostname.

```
mydestination = $myhostname localhost.$mydomain
```

If you want the mail server to receive mail for an entire local network, you need to also specify its domain name. That way, the server can receive mail addressed just to the domain, instead of your specific host.

```
mydestination = $myhostname localhost.$mydomain $mydomain
```

Also, if your host goes by other hostnames and there are DNS records identifying your host by those names, you need to specify those names as well. For example, your host could also be a web server to which mail could be directed. A host **turtle.mytrek.com** may also be identified as the website **mytrek.com**. Both names would have to be listed in the **mydestination** parameter.

```
mydestination = $myhostname localhost.$mydomain $mydomain www.$mydomain
```

If your system is a gateway for one or more local networks, you can specify them with the **mynetworks** parameter. This allows your mail server to relay mail addressed to those networks. Networks are specified using their IP addresses. The **relay_domains** parameter lets you specify domain addresses of networks for which you can relay messages. By default, this is set to **mydestination**:

```
mynetworks=192.168.0.0
relay_domains=$mydestination
```

Hosts within the local network connected to the Internet by a gateway need to know the identity of the relay host, the mail server. You set this with the **relayhost** parameter. Also, **myorigin** should be set to just **mydomain**. If there is a DNS server identifying the gateway as the mail server, you can just set **relayhost** to the value of **mydomain**. If not, then **relayhost** should be set to the specific hostname of the gateway/mail server. If your local network is not running a DNS server, be sure to set **disable_dns_lookups** to **yes**.

```
relay_host=$mydomain
```

Direct Connections

If your system is directly connected to the Internet and you use an ISP (Internet service provider) for receiving mail, you can configure Postfix as a null client to only send mail. Set the **relay_host** parameter to just your own domain name. Also, in the **master.cf** file, comment out the SMTP server and local delivery agent entries.

```
relayhost = $mydomain
```

Masquerading

If your mail server is operating on a gateway for a local network and you want to hide the hosts in that network, you can opt to masquerade the local hosts, letting it appear that all mail is coming from the domain in general, instead of a particular host. To set this option, you use the **masquerade_domains** parameter. In the following example, all mail sent by a local host such as **rabbit.mytrek.com** will be addressed as coming from **mytrek.com**. Thus a message sent by the user **chris@rabbit.mytrek.com** is sent out as coming from **chris@mytrek.com**:

```
masquerade_domains = $mydomain
```

Received mail is not masqueraded by default. This allows Postfix to still deliver received mail to particular hosts. If you want received mail to also be masqueraded, you have to add the **envelope_recipients** parameter to the list of values assigned to the **masquerade_class** parameter. In that case, Postfix will no longer be able to deliver received mail.

Virtual Domains and Virtual Accounts

If your network has implemented virtual domains, you will need to set up a virtual domain table and then specify that table with the **virtual_maps** option. Setting up a table is a simple matter of listing virtual names and their real addresses in a text file such as **/etc/postfix/virtual**. Then use the **postmap** command to create a Postfix table:

```
postmap /etc/postfix/virtual
```

In the **main.cf** file, specify the table with the **virtual_maps** parameter. Postfix will then use this table to look up virtual domains.

```
virtual_maps = hash:/etc/postfix/virtual
```

Note: See the Postfix FAQ at www.postfix.org for detailed information on how to set up Postfix for a gateway, a local workstation, or a host directly connected to the Internet (null server).

Instead of using mail accounts for actual users on a system, you can set up a virtual accounts. Virtual accounts can be managed either in standard Postfix text files, in SQL databases, or as LDAP entries. SQL databases are preferred for managing a large number of virtual accounts. For SQL support, you first create tables in a MySQL database for domains (the virtual domains), users (user accounts), and forwarding (aliases). Corresponding virtual domain configuration files will list information like the database, tables, and host to use, such as a **mysql_virt.cf** for SQL database access and **mysql_users.cf** for accessing the user table. Check the documentation at www.postfix.org for detailed information.

Postfix Greylisting Policy Server

Postfix also supports greylisting with the Postfix Greylisting Policy Server (**postgrey** package). Greylisting blocks spammers based on their mailing methods rather than content, relying on the fact that spammers will not attempt retries if rejected (www.greylisting.org). Messages from new previously unknown sources are rejected, whereupon a valid MTA will retry, whereas a spammer will not. To support the Greylisting Policy Server, Postfix is configured to delegate Policy access to a server. In the **/etc/postfix** directory you can use the **postgrey_whitelist** files to exclude email addresses from greylisting. The Greylisting Policy Server is run as a standalone server, using its own startup script. The postgrey Man page provides detailed information about the server's options.

For more heavy duty work, you can use the SQL Postfix Greylisting Policy Server (**sqlgrey** package). This server uses a SQL database backend for processing greylisting tasks. See http://sqlgrey.sourceforge.net for more details.

Controlling User and Host Access

With an access file, you can control access by certain users, hosts, and domains. The access file works much like the one used for Sendmail. Entries are made in a text file beginning with the user, host, or domain name or address, followed by an action to take. A user, host, or domain can be accepted, rejected with no message, or rejected with a message. Once entries are made, they can be installed in a Postfix database file with the **postmap** command:

```
postmap /etc/postfix/access
```

You can then use the access file in various Postfix operations to control clients, recipients, and senders.

Access can also be controlled by use of the Mail Abuse Prevention System (MAPS), which provides the RBL+ service, a collection of mail address DNS-based databases (**www.mail-abuse.com**). These databases, like the Realtime Blackhole List (RBL), list mail addresses that are known be used by mail abusers. A domain or host is matched against a list maintained by the service, which can be accessed on a local server or directly from an online site. Various Postfix operations let you use MAPS databases to control access by clients, recipients, or senders.

Header and Body Checks

With the **header_checks** parameter, you can specify a Postfix table where you can list criteria for rejecting messages. Check the **/etc/postfix/header_checks** file for details. The criteria are patterns that can match message headers. You can have matching messages rejected, rejected with a reply, simply deleted, or logged with a warning. You have the option of taking several actions, including REJECT, DISCARD, WARN, HOLD, and IGNORE.

```
header_checks = regexp:/etc/postfix/header_checks
```

The database, in this case **/etc/postfix/header_checks**, will have lines, each with a regular expression and a corresponding action. The regular expression can either be a standard regular expression as denoted by **regexp** in the **header_checks** parameter, or conform to a Perl Compatible Regular Expression, **prece**.

The **body_checks** parameter lets you check the body of text messages, line by line, using regular expressions and actions like those used for **header_checks** in an **/etc/postfix/body_checks** file.

Controlling Client, Senders, and Recipients

With the **smtpd_client_restrictions** parameter, you can restrict access to the mail server by certain clients. Restrictions you can apply include **reject_unknown_client_hostname**, which will reject any clients with unresolved addresses; **permit_mynetworks**, which allows access by any clients defined by **mynetworks**; and **check_client_access**, which will check an access database to see if a client should be accepted or rejected. The **reject_rbl_client** and **reject_rhsbl_client** parameters will reject clients from specified domains.

```
smtpd_client_restrictions = permit_mynetworks, \
             reject_unknown_client, check_client_access, reject_maps_rbl
```

The **reject_rbl_client** restriction rejects domain addresses according to a specified MAPS service. The site can be an online site or a local one set up to provide the service. The **reject_rhsbl_client** restriction rejects host addresses.

```
smtpd_client_restrictions = reject_rbl_client relays.mail-abuse.org
```

To implement restrictions from a access file, you can use the **hash** directive and the name of the file.

```
smtpd_client_restrictions = hash:/etc/postfix/access
```

The corresponding **smtpd_sender_restrictions** parameter works much the same way as its client counterpart but controls access from specific senders. It has many of the same restrictions but adds **reject_non_fqdn_sender**, which will reject any mail header without a fully qualified domain name, and **reject_sender_login_mismatch**, which will require sender verification. The **reject_rhsbl_sender** restriction rejects domain addresses according to a specified MAPS service.

The **smtpd_recipient_restrictions** parameter will restrict the recipients the server will accept mail for. Restrictions include **permit_auth_destination**, which allows authorized messages, and **reject_unauth_destination**, which rejects unauthorized messages. The **check_recipient_access** restriction will checks local networks for a recipient address. The **reject_unknown_recipient_domain** restriction rejects recipient addresses with no DNS entry. The **reject_rhsbl_recipient** restriction rejects domain addresses according to a specified MAPS service.

You can further refine restrictions with parameters such as **smtpd_helo_restrictions**, which requires a HELO command from a client. Restriction parameters include **reject_invalid_hostname**, which checks for faulty syntax, **reject_unknown_hostname**, for hosts with no DNS entry, and **reject_non_fqdn_hostname** for hosts whose names are not fully qualified. The **strict_rfc821_envelopes** parameter will implement strict envelope protocol compliance.

Sendmail

Sendmail operates as a server to both receive and send mail messages. Sendmail listens for any mail messages received from other hosts and addressed to users on the network hosts it serves. At the same time, Sendmail handles messages users are sending out to remote users, determining what hosts to send them to. You can learn more about Sendmail at **www.sendmail.org**, including online documentation and current software packages. The Sendmail newsgroup is **comp.mail.sendmail**. You can also obtain a commercial version from **www.sendmail.com**.

The domain name server for your network designates the host that runs the Sendmail server. This is your mail host. Messages are sent to this host, whose Sendmail server then sends the message to the appropriate user and its host. In your domain name server configuration file, the mail host entry is specified with an MX entry. To print the mail queue of messages for future delivery, you can use **mailq** (or **sendmail -v -q**). This runs Sendmail with instructions to print the mail queue.

The Sendmail software package contains several utilities for managing your Sendmail server. These include mailq, which displays the queue of outgoing messages; **mailstats**, which shows statistics on mail server use; **hoststat**, which provides the stats of remote hosts that have connected with the mail server; and **praliases**, which prints out the mail aliases listed in the **/etc/aliases** file. Some, like **mailq** and **hoststat**, simply invoke Sendmail with certain options. Others, like **mailstats** and **praliases**, are separate programs.

File	Description
/etc/mail/sendmail.cf	Sendmail configuration file
/etc/mail/sendmail.mc	Sendmail M4 macro configuration file
/etc/mail/submit.cf	Sendmail configuration file for mail submission mode where Sendmail does not run as a server but merely submits mail.
/etc/mail/submit.mc	Sendmail M4 macro configuration file for Sendmail mail submission mode.
/etc/aliases	Sendmail aliases file for mailing lists
/etc/aliases.db	Sendmail aliases database file generated by the **`newaliases`** command using the aliases file
/etc/mail/access	Sendmail access text file. Access control for screening or relaying messages from different hosts, networks, or users. Used to generate the **access.db** file
/etc/mail/access.db	Sendmail access database file. Generated from the access text file
/etc/mail/local-host-names	Sendmail local hosts file for multiple hosts using the same mail server (formerly **sendmail.cw**)
/etc/mail/trusted-users	Sendmail trusted users file (formerly **sendmail.ct**)
/etc/mail/statistics	Sendmail statistics file (formerly **sendmail.st**)
/etc/mail/virtusertable	Sendmail virtual user table text file. Maps user virtual domain addresses, allowing virtual domains to be hosted on one system. Make entries in this file and then use it to generate the **virtusertable.db** file
/etc/mail/virtusertable.db	Sendmail virtual user table database generated from the **virtusertable** file
/etc/mail/mailertable	Sendmail mailer table text file, used to override routing for your domains
/etc/mail/mailertable.db	Sendmail mailer table database file, generated from the **mailertable** file
/etc/mail/userdb	Sendmail user database file
/etc/mail/domaintable	Sendmail **domaintable** file, maps a domain name to another domain name
/etc/mail/domaintable.db	Sendmail **domaintable** database file, generated from the **domaintable** file
/var/spool/mail	Incoming mail
/var/spool/mqueue	Outgoing mail
/var/spool/maillog	Mail log file

Table 20-2: Sendmail Files and Directories

Sendmail now maintains all configuration and database files in the **/etc/mail** directory. Here you will find the Sendmail macro configuration file, **sendmail.mc**, as well as several database files (see Table 20-2). Many have changed their names with the release of Sendmail 8.10. For example, the help file is now **/etc/mail/helpfile** instead of **/etc/sendmail.ht**. Specialized files provide support for certain features such as **access**, which lets you control access by different hosts and networks to your mail server, and virtusertable, which lets you designate virtual hosts. These files have both text and database versions. The database version ends with the extension **.db** and is the file actually used by Sendmail. You would make your entries in the text version and then effect the changes by generating a corresponding database version. Database versions are generated using the **makemap** command with the **hash** option and a redirection operation for the text and database file. For example, to deny access to a particular host, you would place the appropriate entry for it in the **/etc/mail/access** file, editing the file using any text word processor. Then, to generate the **/etc/mail/access.db** version of the access file, you would change to the **/etc/mail** directory and use the following command:

```
cd /etc/mail
makemap hash access < access
```

To regenerate all the database files, just use the **make** command in the **/etc/mail** directory:

```
make
```

Certain files and directories are used to manage the mail received and sent. Incoming mail is usually kept in the **/var/spool/mail** directory, and outgoing messages are held in the **/var/spool/mqueue** directory, with subdirectories for different users. Monitoring and error messages are logged in the **/var/log/maillog** file.

Note: Red Hat and Fedora place the Sendmail configuration file, **sendmail.cf**, in the **/etc/mail** directory instead of the **/etc** directory as it did in previous versions (7.3 and earlier).

Note: If your mail server services several hosts, you will need to enter them in the **/etc/mail/local-host-names** file.

Aliases and LDAP

Sendmail can now support the Lightweight Directory Access Protocol (LDAP). LDAP enables the use of a separate server to manage Sendmail queries about user mail addresses. Instead of maintaining aliases and **virtusertable** files on different servers, Sendmail uses LDAP support to simply use one centralized LDAP server to locate recipients. Mail addresses are looked up in the LDAP server, instead of having to search several aliases and **virtusertable** files on different servers. LDAP also provides secure authentication of users, allowing controlled access to mail accounts. The following example enables LDAP support on Sendmail in the **sendmail.mc** file:

```
FEATURE('ldap_routing')dnl
LDAPROUTE_DOMAIN('mytrek.com')dnl
```

Alternatively, Sendmail still supports the use of aliases, for either sent or received mail. It checks an aliases database file called **aliases.db** that holds alias names and their associated e-mail addresses. This is often used for administrator mail, where mail may be sent to the system's root user and then redirected to the mail address of the actual system administrator. You can also alias host addresses, enabling you to address hosts on your network using only their aliases. Alias entries

are kept in the **/etc/aliases** file. This file consists of one-line alias records associating aliases with user addresses. You can edit this file to add new entries or to change old ones. They are then stored for lookup in the **aliases.db** file using the command **newaliases**, which runs Sendmail with instructions to update the **aliases.db** file.

Aliases allow you to give different names for an e-mail address or collection of e-mail addresses. One of its most useful features is to create a mailing list of users. Mail addresses to an alias will be sent to the user or list of users associated with the alias. An alias entry consists of an alias name terminated by a colon and followed by a username or a comma-separated list of users. For example, to alias **filmcritic** with the user **george@rabbit.mytrek.com**, you would use the following entry:

```
filmcritic:   george@rabbit.mytrek.com
```

To alias **singers** with the local users **aleina** and **larisa**, you would use

```
singers:     aleina, larisa
```

You can also use aliases as the target addresses, in which case they will expand to their respective user addresses. For example, the **performers** alias will expand through the **filmcritic** and **singers** aliases to the users **george@rabbit.mytrek.com**, **aleina**, and **larisa**:

```
performers:     filmcritic, singers
```

Once you have made your entries in the **/etc/mail/aliases** file, you need to generate a database version using the **newaliases** command:

```
newaliases
```

Note: Sendmail now supports a mail submission mode. In this mode, Sendmail does not run as a server along with the corresponding root privileges. Instead it merely submits mail. In this mode, Sendmail can be invoked directly by Email clients to send mail. The mail submission mode uses the **/etc/mail/submit.cf** configuration file which can be configured using macros in the **/etc/mail.submit.mc** file.

Sendmail Configuration

The main Sendmail configuration file is **sendmail.cf**, located in the **/etc** directory. This file consists of a sometimes lengthy list of mail definitions that set general options, designate MTAs, and define the address rewrite rules. A series of options set features, such as the maximum size of mail messages or the name of host files. The MTAs are those mailers through which Sendmail routes messages. The rewrite rules "rewrite" a mail address to route through the appropriate Internet connections to its destination (these rules can be complex). Check the Sendmail HOW-TO and the online documentation for a detailed explanation.

The **sendmail.cf** definitions can be complex and confusing. To simplify the configuration process, Sendmail supports the use of macros you can use to generate the **sendmail.cf** file using the M4 preprocessor (this requires installation of the sendmail-cf package). Macros are placed in the **/etc/mail/sendmail.mc** file. Here, you can use macros to designate the definitions and features you want for Sendmail, and then the macros are used to generate the appropriate definitions and rewrite rules in the **sendmail.cf** file. As part of the Sendmail package, several specialized versions of the **sendmail.mc** file are made available in the **/usr/share/sendmail-cf** directory. These begin with a

system name and have the suffix **.mc**. On many distributions, a specialized version tailored to your distribution is already installed as your **/etc/mail/sendmail.mc** file.

Once you configure your **sendmail.mc** file, you use the following command to generate a **sendmail.cf** file (be sure first to back up your original **sendmail.cf** file). You can rename the **sendmail.mc** file to reflect the specific configuration. You can have as many different **.mc** files as you want and use them to implement different configurations. On Red Hat and Fedora you can use the following command in the **/etc/mail** directory:

```
make -C /etc/mail
```

Alternatively, you can use the original **m4** macro command:

```
m4 sendmail.mc > /etc/mail/sendmail.cf
```

You will then need to restart the Sendmail server to make the configuration effective:

```
service sendmail restart
```

In the **sendmail.mc** file, you configure different aspects of Sendmail using either a **define** command to set the value of Sendmail variables or a Sendmail macro that has already been defined to set a particular Sendmail feature. For example, to assign the **PROCMAIL_PATH** variable to the directory **/usr/bin/procmail**, you would use the following:

```
define('PROCMAIL_MAILER_PATH','/usr/bin/procmail')
```

Similarly, if there are variables that you do not want defined, you can remove them with the **undefine** command:

```
undefine('UUCP_RELAY')
```

To specify the type of operating system that your Sendmail server is running on, you would use the **OSTYPE** Sendmail macro. The following example specifies the Linux operating system:

```
OSTYPE('linux')
```

Feature	Description
use_cw_file	Checks for hosts served by the mail server **/etc/mail/local-host-names** file.
use_ct_file	Reads a list of users from the **/etc/trusted-users** file. These are trusted users that can change the sender name for their messages.
redirect	Rejects all mail addressed to "address.REDIRECT", providing a forwarding address is placed in the **/etc/aliases** file.
mailertable	Uses a mailer table file, **/etc/mail/mailertable**, to override routing for particular domains.
domaintable	Uses a domain table file, **/etc/mail/domaintable,** to map one domain to another. Useful if you change your domain name.

allmasquerade	Causes recipient addresses to also masquerade as being from the masquerade host.
masquerade_entire_domain	Masquerades all hosts within the domain specified in **MASQUERADE_AS**.
masquerade_envelope	Masquerades envelope sender and recipient along with headers.
virtusertable	For virtual hosts, maps virtual addresses to real addresses.
nullclient	Turns a Sendmail server into a null client, which simply forwards mail messages to a central mail server for processing.
local_procmail	Uses procmail as the local mailer.
smrsh	Uses the Sendmail Restricted Shell (smrsh) for mailing.
promiscuous_relay	Allows you to relay mail, allowing mail to be received from outside your domain and sent on to hosts outside your domain.
relay_entire_domain	Allows any host in your domain to relay mail (default limits this to hosts in the access database).
relay_hosts_only	Checks for relay permission for particular hosts instead of domains.
accept_unqualified_senders	Allows sender e-mail addresses to be single usernames instead of just fully qualified names that include domain names.
accept_unresolvable_domains	Allows Sendmail to accept unresolvable domain names. Useful for those users in a local network blocked by a firewall from the full DNS namespace. By default, Sendmail requires domains in addresses to be resolvable with DNS.
access_db	Accepts or rejects mail from domains and hosts in the access database.
blacklist_recipients	Blocks mail to certain users, such as those that should never receive mail—like the users **nobody** and **host**.
dnsbl	Rejects hosts in the Realtime Blackhole List. Managed by MAPS (Mail Abuse Prevention System LLC) and designed to limit transport of unwanted mass e-mail (www.mail-abuse.org).
ldap_routing	Enables LDAP use.

Table 20-3: Sendmail Features

The **MAILER** macro specifies the mail delivery agents (MDAs) to be used. You may have more than one. Usually, you will need a mail delivery agent such as procmail for delivering mail to hosts on your network. In addition, Sendmail in effect operates as an MDA to receive messages from hosts in its local network, which it will then send out to the larger network.

```
MAILER(procmail)
MAILER(smtp)
```

Sendmail also supports an extensive number of features that you need to explicitly turn on. You can do this with the Sendmail **FEATURE** macro. See Table 20-3 for a list of commonly used Sendmail features. The following example turns on the **redirect** feature, which is used to inform a sender that a recipient is now at a different address:

```
FEATURE(redirect)
```

In addition, you can set certain configuration options. These are variables beginning with the prefix **conf** that you can set and assign values to using the **define** command. There are an extensive number of configuration options, most of which you will not need to change. The following example defines the **confAUTO_REBUILD** configuration option, which will automatically rebuild the aliases database if needed:

```
define('confAUTO_REBUILD')
```

Certain macros and types of macros need to be placed in the **sendmail.mc** file in a particular sequence as shown here. Notice that **MAILER** is toward the end and **OSTYPE** at the beginning. Local macro definitions (**define**) and **FEATURE** entries follow the **OSTYPE** and **DOMAIN** entries:

```
VERSIONID
OSTYPE
DOMAIN
define
FEATURE
local macro definitions
MAILER
LOCAL_RULE_*
LOCAL_RULESETS
```

The local macro and configuration option definitions that affect a particular feature need to be entered before the **FEATURE** entry. For example, the **redirect** feature uses the aliases file. Any local definition of the aliases file needs to be entered before the **redirect** feature.

```
define('ALIAS_FILE','/etc/aliases')
FEATURE(redirect)
```

You need to be careful how you enter comments into a **sendmail.mc** file. This file is read as a stream of macros, ignoring all white spaces, including newlines. No special comment characters are looked for. Instead, you have to simulate comment indicators using the **dnl** or **divert** commands. The **dnl** command instructs that all characters following that **dnl** command up to and including the next newline are to be ignored. If you place a **dnl** command at the beginning of a text line in the **sendmail.mc** file, it has the effect of turning that line into a comment, ignoring everything on that line—including its newline. Even empty lines will require a **dnl** entry to ignore the newline character:

```
dnl you will have to /etc/mail/sendmail.cf by running this the m4
dnl macro config through preprocessor:
dnl
```

Alternatively, you can use the **divert** command. The **divert** command will ignore all data until another **divert** command is reached:

```
divert(-1)
 This is the macro config file used to generate
 the /etc/mail/sendmail.cf file. If you modify the file regenerate
 you will have to regenerate /etc/mail/sendmail.cf by running the m4
 macro
divert(0)
```

For Sendmail to work at all, it requires only that the **OSTYPE** and **MAILERS** macros be defined, along with any needed features and options. A very simple Sendmail file is shown here:

mysendmail.mc

```
dnl My sendmail.mc file
OSTYPE('linux')
define('PROCMAIL_MAILER_PATH','/usr/bin/procmail')
FEATURE(redirect)
MAILER(procmail)
MAILER(smtp)
```

A **sendmail.mc** file usually contains many more entries, particularly for parameters and features. Check the **/etc/mail/sendmail.mc** file on your Red Hat or Fedora system to see the standard default entries for Sendmail.

Sendmail Masquerading

For a mail server that is relaying messages from local hosts to the Internet, you may want to masquerade the source of the messages. In large networks that have their own mail servers connected to the Internet, Sendmail masquerading can make messages sent by local hosts appear to be sent by the mail server. Their host address will be replaced by the mail server's address. Returned mail can then be sent to the mail server and held in POP or IMAP server mailboxes that can be later accessed by users on the local hosts. Also, entries in the server's virtual user table could forward mail to corresponding users in local hosts.

Masquerading is often used to mask local hosts with a domain name. Any subdomains can also be masqueraded. This method can be applied to situations where an ISP or your network administrator has assigned your network its own domain name. You can then mask all mail messages as coming from your domain name instead of from particular hosts or from any subdomains you may have. For example, if a network's official domain name is **mytrek.com**, all messages from the hosts in the **mytrek.com** network, such as **rabbit.mytrek.com** and **turtle.mytrek.com**, could be masqueraded to appear as just coming from **mytrek.com**. Should the **mytrek.com** network have a subnetwork whose domain is **mybeach.com**, any messages from **mybeach.com** could also be masqueraded as coming from **mytrek.com**.

Masquerading is turned on with the **MASQUERADE_AS** command. This takes as its argument the name you want to masquerade your mail as. Normally, the name used is just the domain name, without the mail host. In the following example, the mail is masqueraded as simply **mytrek.com**. Mail sent from a local host like **turtle.mytrek.com** will appear to be sent by just **mytrek.com**:

```
MASQUERADE_AS('mytrek.com')dnl
```

You will also have to specify the hosts and domains on your local network that your Sendmail server should masquerade. If you have decided to masquerade all the hosts in your local network, you just need to set the **masquerade_entire_domain** feature, as in

```
FEATURE('masquerade_entire_domain')dnl
```

If, instead, you want to masquerade particular hosts or your domain has several subdomains that you want masqueraded, you list them in the **MASQUERADE_DOMAIN** entry. You can list either particular hosts or entire domains. For example, given a local network with the local hosts **turtle.mytrek.com** and **rabbit.mytrek.com**, you can list them with the **MASQUERADE_DOMAIN** to have them masqueraded. The domain they are masqueraded as is specified in the **MASQUERADE_AS** entry.

```
MASQUERADE_DOMAIN('turtle.mytrek.com rabbit.mytrek.com')dnl
```

If you want to masquerade all the hosts in your local network, you can simply list your local network's domain name. If your local network also supports several subdomains, you can list those as well to masquerade them. For example, to masquerade all the hosts in the **mybeach.com** domain, you would use the following entry:

```
MASQUERADE_DOMAIN('mytrek.com mybeach.com')dnl
```

If you have a long list of domains or hosts, or if you want to be able to easily change those that should be masqueraded, you can place them in a file to be read by Sendmail. Specify the file with the **MASQUERADE_DOMAIN_FILE** command:

```
MASQUERADE_DOMAIN_FILE('mydomains')dnl
```

If you just want to masquerade all the hosts in your local domain, you use the **masquerade_entire_domain** feature:

```
FEATURE(masquerade_entire_domain)dnl
```

A common configuration for a local network would specify the domain name in the **MASQUERADE_AS** entry and in the **MASQUERADE_DOMAIN** entry. Using the example **myisp.com** for the domain, the entries would look like this:

```
MASQUERADE_AS('mytrek.com')dnl
FEATURE(masquerade_entire_domain)dnl
```

If you wanted to masquerade as an ISP's mail domain, you would use the ISP's domain in the **MASQUERADE_AS** entry as shown here:

```
MASQUERADE_AS('myisp.com')dnl
MASQUERADE_DOMAIN('mytrek.com')dnl
```

When mail is received from the outside bearing just the address **mytrek.com**, your network needs to know what host to send it to. This is the host designated as the mail server for the **mytrek.com** network. This information is provided by a mail exchange record (MX) in your DNS configuration that will specify that mail sent to **mytrek.com** will be handled by the mail server—in this case, **turtle.mytrek.com**:

```
mytrek.com.  IN    MX     0     turtle.mytrek.com.
```

You further have to be sure that MX relaying is enabled with the **relay_based_on_MX** feature:

```
FEATURE(relay_based_on_MX)dnl
```

All messages will appear to originate from the mail server's host. For example, if your Sendmail mail server is running on **turtle.mytrek.com**, mail sent from a local host called **rabbit.mytrek.com** will appear to have been sent from **turtle.mytrek.com**.

You can also masquerade recipient addresses, so that mail sent to users on your local host will be sent instead to the masqueraded address. Use the **allmasquerade** feature to enable recipient masquerading:

```
FEATURE(allmasquerade)dnl
```

Configuring Mail Servers and Mail Clients

Sendmail can be used either as a mail server, handling mail for various hosts on a network, or as a mail client, managing mail for local users on a particular host. In a simple network configuration, you would have each host running Sendmail in a client configuration, and one host operating as a mail server, relaying mail for the network hosts. For a local network connected to the Internet, your local hosts would run Sendmail in a client configuration, and your gateway would run Sendmail in a server configuration (though the mail server would not have to necessarily run on the gateway). The mail server would relay messages from the local network hosts out to the Internet. The mail server could also be used to block unwanted access from outside hosts, such as those sending spam mail. A basic client or server Sendmail configuration involves just a few features in the **/etc/mail/sendmail.mc** file. The default Fedora configuration installed on your system allows use on a single host, managing messages between users on that host. To enable client and server use, you will need to make changes to the **/etc/mail/sendmail.mc** file.

Configuring Sendmail for a Simple Network Configuration

Fedora initially configures Sendmail to work only on the system it is running on, **localhost**. To use Sendmail to send messages to other hosts on a local network, you need to change and add settings in the **sendmail.mc** and **/etc/mail/access** files. A simple network configuration would have Sendmail running on each host, handling both mail sent between users on that host and mail to and from users on other hosts. For each Sendmail server configuration, you would make the changes described in the following section on simple local network configuration.

For messages sent between hosts on your network, you only need to run the Sendmail server on each, making a few changes to their Sendmail configurations. The Sendmail server on one of your hosts can be configured to handle the task of relaying messages between hosts. Using the network example described earlier, the hosts **turtle**, **rabbit**, and **lizard** will be running their own Sendmail servers. The Sendmail server on the **turtle** host will be configured to relay messages between all the host, itself included.

On each host on your network, edit the **/etc/mail/sendmail.mc** file and make the following changes. Comment out the **DAEMON_OPTIONS** line in the default **sendmail.mc** file by placing a **dnl** word in front of it, as shown here. Removing this feature will allow you to receive messages over your local network. This entry is restricting Sendmail to the **localhost** (127.0.0.1):

```
dnl DAEMON_OPTIONS('Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

In the **sendmail.mc** file located on the host that you want to have handle the relaying of messages, you need to also add the following line:

```
FEATURE(relay_entire_domain)dnl
```

Run the **m4** operation to install the changed configuration and then restart the server with the service operation, as described earlier.

You can now e-mail messages from one user to another across your network. For example, **george@turtle.mytrek.com** can now e-mail a message to **larisa@rabbit.mytrek.com**. The local Sendmail servers will take care of sending and delivering mail both to users within their hosts and those located on other network hosts.

Configuring Sendmail for a Centralized Mail Server

Alternatively, you could set up a central mail server to handle all the mail on your network. Mail clients on various hosts could send their messages to the central mail server, which would then relay them out to the larger network or Internet. Mail could then be received at the central mail server, where clients could later retrieve it. There are several ways to set up a central mail server. One of the simplest is to run a central mail server on your gateway host, and then have null client versions of the Sendmail server running on local hosts. Any mail sent from local hosts would be automatically forwarded to the central mail server. Received mail could only be delivered to the central server, usually to a POP or IMAP server also running on the central server's host. Users could then access the POP server to retrieve their mail.

For a centralized configuration, it would make sense to treat users as having their network domain as their address, rather than separate hosts in their network. Thus the user **cece** on **rabbit.mytrek.com** would use the mail address **cece@mytrek.com**, not **cece@rabbit.mytrek.com**. Users could have the same name as those on their respective hosts, but corresponding users would be set up on the gateway host to handle received mail managed by the POP or IMAP servers.

An effective simple mail server would involve several components:

- A central mail server running on the gateway host
- Each client running Sendmail as a null client
- Masquerading all mail to use the domain address only, not host addresses
- A POP or IMAP server running on the gateway host to handle received mail

Configuring a Workstation with Direct ISP Connection

If you are running a Linux system that is not part of a network but does have a direct connection to the Internet through an ISP (Internet service provider), you could simply use the ISP mail servers for sending and receiving mail. Normally, you would have an SMTP mail server for outgoing mail and a POP server for incoming mail. However, you can also configure Sendmail to interface with your ISP.

Be sure to first comment out the **DAEMON_OPTIONS** option as shown in the previous sections.

Normally, your ISP will provide a mail server that will handle mail for its hosts. To make use of the ISP mail server, you can define it with the **SMART_HOST** option. Mail will be sent through the ISP mail server. **SMART_HOST** has the format *type*:*hostname,* where *type* is the kind of

mail server used, usually SMTP. The default is relay. Define the **SMART_HOST** option to use your ISP to send and receive mail:

```
define ('SMART_HOST', 'smtp:mail.my-isp.com')dnl
```

The **SMART_HOST** option is used to indicate a specific remote mail server that you want to have handled the relaying of your network messages. It can be an ISP mail server, as well as any mail server in a larger network.

For a dial-up connection over a modem, you can use various configuration options to control your connection. The **confMESSAGE_TIMEOUT** option lets you control how long mail can remain on the output queue, letting you keep mail until you are ready to dial in and send it. Setting the **confDELIVERY_MODE** option to **queueonly** lets you send mail only when you are ready.

The Mailer Table

The mailer table lets you route messages addressed a specified host or domain to particular mail server. You can use the mailer table to have mail addressed to a virtual domain routed to the mail server for your network. To reference an entire domain, prefix the domain name with a period. The host to which the mail is routed is prefixed by the mailer used, usually **smtp** for Sendmail. The following entry will route mail addressed to **.mybeach.com** to the mail server **turtle.mytrek.com**:

```
.mybeach.com       smtp:turtle.mytrek.com
```

Entries are placed in the **/etc/mail/mailertable** file. Once you have made your entries, generate the **mailertable.db** database file with the **make** command:

```
make mailertable
```

Virtual Domains: virtusertable

You can define virtual domains for your network. These virtual domains are mapped to one or more real domains by your DNS server. However, you can receive messages with mail addresses for users on your virtual domains. In this case, you need to map these addresses to users on your real domain so that the mail can be delivered to an existing location. This mapping is carried out by the virtual user table called **/etc/mail/virtusertable**. The virtual user table lets you map mail addresses for virtual domains to users on real domains. Once you have made your entries, generate the **virtusertable.db** database file with the **make** command:

```
make virtusertable
```

Security

For security, Sendmail lets you screen specific messages as well as provide authentication and encryption for Sendmail transmissions. With version 8.11, Sendmail incorporated support for the Secure Sockets Layer (SSL) and the Simple Authentication and Security Layer (SASL). Support for SSL goes by the Sendmail command **STARTTLS**, which stands for "start transport layer security." SSL provides authentication, encryption, and integrity checks for Sendmail operations. OpenSSL must first be installed to allow use of SSL encryption and authentication methods.

The SASL is implemented by the **AUTH** command and is referred to as SMTP AUTH. SASL provides authentication for mail users and servers. It can make use of already-installed Kerberos services to provide authentication.

Sendmail also provides you with the capability of screening out messages from specific domain, host, IP, and user addresses. Rules to perform such screening are kept in the **/etc/mail/access** file. You can edit this file and add your own rules. A rule consists of an address followed by an action to take. (The actions supported are listed in Table 20-4.) For example, to remove all messages from the **myannoyingad.com** domain, you would enter

```
myannoyingad.com DISCARD
```

The next example rejects any message from **larisa@turtle.mycar.com** and sends a notice of the rejection:

```
larisa@turtle.mycar.com REJECT
```

You can also specify an error message to return, as shown here:

```
cecelia@rabbit.mytrek.com    ERROR:”Retired yesterday”
```

To send an error message to spammers, you could include a message as shown here. The first number is an error code:

```
cyberspammer.com     ERROR:"550 We don't accept mail from spammers"
```

An **/etc/mail/access** file with the previous entries would look like the following:

```
myannoyingad.com           DISCARD
larisa@turtle.mycar.com    REJECT
cecelia@rabbit.mytrek.com  ERROR:"Retired yesterday"
cyberspammer.com           ERROR:"550 We don't accept mail from spammers"
```

Sendmail actually reads the access rules from a database file called **access.db**, also located in the **/etc/mail** directory. To implement your rules, you have to regenerate the **access.db** file using the access file. You can do this with the **make** command using **access** as the argument, as shown here:

```
make access
```

Sendmail then has to be restarted to read the new **access.db** file.

The use of the access file is enabled in the **sendmail.mc** file with the **access_db** feature:

```
FEATURE('access_db')dnl
```

The access file will deny mail received from the listed addresses. However, you can also reject any mail sent to them. Additionally, you can also received mail for certain hosts on your network. You do this by enabling the **blacklist_recipients** option in the **sendmail.mc** file. This option governs recipients, whereas **access** normally governs senders. Those addresses listed will not be able to receive any mail. This feature is also used for certain administrative users that should never receive mail, such as **nobody** (the guest user) or **ftp** (the FTP user):

```
FEATURE('blacklist_recipients')dnl
```

The following example will not allow mail to be sent to **cyberspammer.com** (a recipient), nor can mail be received for **justin@lizard.mytrek.com**, **secretproject@rabbit.mytrek.com**, or **mysurfboard.com**:

```
mysurfboard.com                ERROR:"Domain does not exist"
justin@lizard.mytrek.com       "Moved to Hawaii"
```

```
secretproject@rabbit.mytrek.com  REJECT
cyberspammer.com                 REJECT
```

Your distribution version of **smb.conf** may configure Sendmail to use **access_db** (as is the case with Fedora). Access is granted only to users on the local host. If your system is being used as a mail server for a network and you have not enabled the **relay_entire_domain** feature, you will need to allow access by other hosts on your network. In the access file, you can place a **RELAY** rule for your network. The **RELAY** rule will let other hosts use your mail server to send messages out to other hosts. This is normally done for a gateway host that needs to relay messages from a local network out to the Internet. The following example allows access from the **mytrek.com** network:

```
mytrek.com      RELAY
```

For a specific host, place an entry for it in the access file as shown here:

```
rabbit.mytrek.com    RELAY
```

To further secure Sendmail, you should disable the use of **VRFY**. This option allows remote users to try to verify the existence of a user address. This can be used to guess valid users on your system. This option is disabled with the **noverify** feature:

```
FEATURE('noverify')dnl
```

Another potential security breach is the **EXPN** option, which expands mailing lists and aliases to their actual addresses. Use the **noexpn** feature to turn it off:

```
FEATURE('noexpn')dnl
```

By default, Sendmail will refuse mail from any domain that cannot be resolved. You can override this restriction with the **accept_unresolvable_domains** feature. Sendmail will also reject mail whose addresses do not have fully qualified domain names. You can override this feature with **accept_unqualified_senders**.

Action	Description
OK	Accepts message even if other rules would reject (exception to the rules).
DISCARD	Discards the message completely.
REJECT	Rejects the message, sending a rejection notice to the sender.
RELAY	Relays messages for specified domain.
SMTP-code message	Code and message to be sent to sender.

Table 20-4: Access Actions

POP and IMAP Server: Dovecot

The protocols Internet Mail Access Protocol (IMAP) and Post Office Protocol (POP) allow a remote server to hold mail for users who can then fetch their mail from it when they are ready. Unlike procmail, which delivers mail messages directly to a user account on a Linux system, the IMAP and POP protocols hold mail until a user accesses an account on the IMAP or POP server. The servers then transfer any received messages to the user's local mailbox. Such servers are often used by ISPs to provide Internet mail services for users. Instead of being sent directly to a

user's machine, the mail resides in the IMAP or POP server until it's retrieved. Fedora can install Dovecot as its IMAP and POP servers. Other popular IMAP and POP servers available are Qpopper, the Qmail POP server, the Washington University POP and IMAP servers, and the Courier POP and IMAP servers.

You can access the POP server from different hosts; however, when you do, all the messages are transferred to that host. They are not kept on the POP server (though you can set an option to keep them). The POP server simply forwards your messages on the requesting host. When you access your messages from a certain computer, they will be transferred to that computer and erased from the POP server. If you access your POP server again from a different computer, those previous messages will be gone.

The Internet Mail Access Protocol (IMAP) allows a remote server to hold mail for users who can then log in to access their mail. Unlike the POP servers, IMAP servers retain user mail messages. Users can even save their mail on the IMAP mail server. This has the advantage of keeping a user's mail in one centralized location accessible anywhere on the network. Users can log in to the mail server from any host on the network and read, send, and save their mail.

Unlike POP, IMAP allows users to set up multiple folders on their mail server in which they can organize their mail. IMAP also supports the use of shared folders to which several users can access mail on a given topic.

Dovecot

Dovecot is a combination IMAP and POP server. Using its own indexing methods, Dovecot is able to handle a great deal of e-mail traffic. It features support for SSL, along with numerous authentication methods. Password database support includes shadow passwords, LDAP, PAM, and MySQL. Dovecot is available in POP, IMAP, and common packages on the Fedora main repository. The **/etc/dovecot.conf** file is configured to use plain password authentication with PAM, using the **passwd** file.

Configuration options are placed in **/etc/dovecot.conf**. This file contains commented default settings with detail explanations for each. Option specific to **imap** and **pop3** are placed in their own sections. These are some basic settings to configure:

- **protocols** This can be set to `imap` and `pop`, as well as `imaps` and `pops` for SSL-encrypted connections.
- **listen** These can be set to IPv4 or IPv4 protocols; IPv6 is set by default. The listen option is set in the respective protocol sections, like `protocol imap` or `protocol pop3`.
- **auth default** section This section holds your default authentication options.
- **mechanism** in **auth** section plain by default digest-MD5 and cran-MD5 are supported, but they are not needed if you are using SSL.
- **passwd** in **auth** section**mail_location** The default mail storage method and location.

Dovecot supports either mailbox or maildir (IMAP) storage formats. The mailbox format uses single large mailbox files to hold several mail messages. Updates can be time consuming. The maildir format uses a separate file for each message, making updates much more efficient. Dovecot will automatically detect the kind of storage use, referencing the MAIL environment variable. This will be the user's **mbox** file at **/var/mail**. You can configure Dovecot to use a maildir format by

setting the **mail_location** option to use a maildir setting, specifying the directory to use. The **%u** symbol can be used to represent the user name, **%h** for the home directory. Messages will be stored in a user's **maildir** directory instead of an **mbox** file. Be sure to create the **maildir** directory and give it read, write, and execute access.

```
mail_location=maildir:/var/mail/%1u/%u/maildir
```

Other POP and IMAP Servers

Many distributions also include the Cyrus IMAP server, which you can install and use instead of Dovecot. In addition, several other IMAP and POP servers are available for use on Linux:

- The University of Washington POP and IMAP servers (**ftp.cac.washington.edu/imap**) are part of the University of Washington's **imap** package. The POP server daemons are called **ipop2d** and **ipop3d**. Your Linux system then runs as a POP2 and POP3 server for your network. These servers are run through **xinetd**. The POP3 server uses the **ipop3** file in the **/etc/xinetd.d**, and the IMAP uses **imap**.
- The Cyrus IMAP server (**http://cyrusimap.web.cmu.edu**) features security controls and authentication, using a private mailbox structure that is easily. Designed to be run on dedicated mail servers, it is supported and maintained by Carnegie Mellon. The name of the Cyrus IMAP server daemon is **imapd**. There will be a file called **imap** in the **/etc/xinetd.d** directory.
- The Courier-IMAP server (**http://courier-mta.org**) is a small, fast IMAP server that provides extensive authentication support including LDAP and PAM.
- Qpopper is the Berkeley POP server (popper). Qpopper is unsupported software, currently available from Qualcomm, makers of Eudora e-mail software. The Qpopper web page is located at the Eudora site archives (**www.eudora.com**).

Note: The IMAP and POP servers provide SSL encryption for secure e-mail transmissions. You can also run IMAP and POP servers using Stunnel to provide similar security. Stunnel is an SSL wrapper for daemons like **imapd**, **popd**, and even **pppd** (modem connections). In the service's **xinetd** script, you can invoke the server with the **stunnel** command instead of running the server directly.

Spam: SpamAssassin

With SpamAssassin, you can filter sent and received e-mail for spam. The filter examines both headers and content, drawing on rules designed to detect common spam messages. When they are detected, it then tags the message as spam, so that a mail client can then discard it. SpamAssassin will also report spam messages to spam detection databases. The version of SpamAssassin distributed for Linux is the open source version developed by the Apache project, located at **http://spamassassin.apache.org**. There you can find detailed documentation, FAQs, mailing lists, and even a listing of the tests that SpamAssassin performs.

SpamAssassin rule files are located at **/usr/share/spamassassin**. The files contain rules for running tests such as detecting the fake hello in the header. Configuration files for SpamAssassin are located at **/etc/mail/spamassassin**. The **local.cf** file lists system-wide

SpamAssassin options such as how to rewrite headers. The **init.pre** file holds spam system configurations. The **spamassassin-spamc.rc** file will redirect all mail to the **spamc** client.

Users can set their own SpamAssassin option in their **.spamassassin/user_prefs** file. Common options include **required_scorei**, which sets a threshold for classifying a message as SPAM, numerous whitelist and blacklist options that accept and reject messages from certain users and domains, and tagging options that either rewrite or just add SPAM labels. Check the Mail::SpamAssassin::Conf Man page for details.

To configure procmail to use SpamAssassin, you need to have procmail run the **/etc/mail/spamassassin/spamassassin-spamc.rc** file. This will filter all mail through SpamAssassin. The **spamassassin-spamc.rc** file uses the **spamd** daemon, which means you have to have the SpamAssassin service running. The **spamassassin-default.rc** file runs a less efficient script to use SpamAssassin, instead of the daemon. If you want system wide procmail filtering, you use the **/etc/procmailrc** file; whereas to implement filtering on a per-user basis, you use a **.procmail** file in the user's home directory. Within the respective procmail files, add the following at the top:

```
INCLUDERC=/etc/mail/spamassassin/spamassassin-spamc.rc
```

Configuring Postfix for use with SpamAssassin can be complicated. A helpful tool for this task is **amavisd-new**, an interface between a mail transport agent like Exim or Postfix and content checkers like SpamAssassin and virus checkers. Check **ijs.si/software/amavisd** for more details.

21. FTP

The File Transfer Protocol (FTP) is designed to transfer large files across a network from one system to another. Like most Internet operations, FTP works on a client/server model. FTP client programs can enable users to transfer files to and from a remote system running an FTP server program. Any Linux system can operate as an FTP server. It has to run only the server software—an FTP daemon with the appropriate configuration. Transfers are made between user accounts on client and server systems. A user on the remote system has to log in to an account on a server and can then transfer files to and from that account's directories only. A special kind of user account, named *ftp,* allows any user to log in to it with the username "anonymous." This account has its own set of directories and files that are considered public, available to anyone on the network who wants to download them. The numerous FTP sites on the Internet are FTP servers supporting FTP user accounts with anonymous login. Any Linux system can be configured to support anonymous FTP access, turning them into network FTP sites. Such sites can work on an intranet or on the Internet.

FTP Servers

FTP server software consists of an FTP daemon and configuration files. The *daemon* is a program that continuously checks for FTP requests from remote users. When a request is received, it manages a login, sets up the connection to the requested user account, and executes any FTP commands the remote user sends. For anonymous FTP access, the FTP daemon allows the remote user to log in to the FTP account using **anonymous** or **ftp** as the username. The user then has access to the directories and files set up for the FTP account. As a further security measure, however, the daemon changes the root directory for that session to be the FTP home directory. This hides the rest of the system from the remote user. Normally, any user on a system can move around to any directories open to him or her. A user logging in with anonymous FTP can see only the FTP home directory and its subdirectories. The remainder of the system is hidden from that user. This effect is achieved by the `chroot` operation (discussed later) that literally changes the system root directory for that user to that of the FTP directory. By default, the FTP server also requires a user be using a valid shell. It checks for a list of valid shells in the **/etc/shells** file. Most daemons have options for turning off this feature.

FTP Servers	Site
Very Secure FTP Server (vsftpd)	http://vsftpd.beasts.org
ProFTPD	www.proftpd.org
PureFTP	www.pureftpd.org

Table 21-1: FTP Servers

Available Servers

Several FTP servers are available for use on Linux systems (see Table 21-1). Three of the more common servers include **vsftpd**, **pureftpd**, and **proftpd**. The Very Secure FTP Server provides a simple and very secure FTP server. The Pure FTPD servers is a lightweight, fast, and secure FTP server, based upon Troll-FTPd. Documentation and the latest sources are available from www.pureftpd.org.

ProFTPD is a popular FTP daemon based on an Apache web server design. It features simplified configuration and support for virtual FTP hosts. The compressed archive of the most up-

to-date version, along with documentation, is available at the ProFTPD website at www.proftpd.org.

FTP Users

Normal users with accounts on an FTP server can gain full FTP access simply by logging into their accounts. Such users can access and transfer files directly from their own accounts or any directories they may have access to. You can also create users, known as guest users that have restricted access to the FTP publicly accessible directories. This involves setting standard user restrictions, with the FTP public directory as their home directory. Users can also log in as anonymous users, allowing anyone on the network or Internet to access files on an FTP server.

Anonymous FTP: vsftpd

An anonymous FTP site is essentially a special kind of user on your system with publicly accessible directories and files in its home directory. Anyone can log in to this account and access its files. Because anyone can log in to an anonymous FTP account, you must be careful to restrict a remote FTP user to only the files on that anonymous FTP directory. Normally, a user's files are interconnected to the entire file structure of your system. Normal users have write access that lets them create or delete files and directories. The anonymous FTP files and directories can be configured in such a way that the rest of the file system is hidden from them and remote users are given only read access. In ProFTPD, this is achieved through configuration directives placed in its configuration file. An older approach implemented by the vsftpd package involves having copies of certain system configuration, command, and library files placed within subdirectories of the FTP home directory. Restrictions placed on those subdirectories then control access by other users. Within the FTP home directory, you then have a publicly accessible directory that holds the files you want to make available to remote users. This directory usually has the name **pub**, for public.

An FTP site is made up of an FTP user account, an FTP home directory, and certain copies of system directories containing selected configuration and support files. Newer FTP daemons, such as ProFTPD, do not need the system directories and support files. Most distributions have already set up an FTP user account when you installed your system.

The FTP User Account: anonymous

To allow anonymous FTP access by other users to your system, you must have a user account named *FTP*. Most distributions already create this account for you. If your system does not have such an account, you will have to create one. You can then place restrictions on the FTP account to keep any remote FTP users from accessing any other part of your system. The entry for this account in your **/etc/passwd** file must be set up to prevent normal user access to it. The following is the entry you find in your **/etc/passwd** file on Fedora that sets up an FTP login as an anonymous user:

```
ftp:x:117:65534::/home/ftp:
```

The **x** in the password field blocks the account, which prevents any other users from gaining access to it, thereby gaining control over its files or access to other parts of your system. The user ID, 14, is a unique ID. The comment field is FTP User. The login directory is **/home/ftp**. When FTP users log in to your system, they are placed in this directory. If a home directory has not been set up, create one and then change its ownership to the FTP user with the **chown** command.

FTP Group

The group ID is the ID of the **ftp** group, which is set up only for anonymous FTP users. You can set up restrictions on the **ftp** group, thereby restricting any anonymous FTP users. Here is the entry for the **ftp** group you find in the **/etc/group** file. If your system does not have one, you should add it:

```
ftp::117:
```

Creating New FTP Users

If you are creating virtual FTP hosts, you will need to create an FTP user for each one, along with its directories. For example, to create an FTP server for a host1-ftp host, you create a host1-ftp user with its own directory.

```
useradd -d /home/host1-ftp host1-ftp
```

This creates a user such with a password entry as that described here:

```
host1-ftp:x:118:50::/home/host1-ftp:
```

You also need to create the corresponding home directory, **/home/host1-ftp** in this example, and set its permissions to give users restricted access.

```
mkdir /home/host1-ftp
chmod 755 /home/host1-ftp
```

In addition, you need to make sure that the root user owns the directory, not the new FTP users. This gives control of the directory only to the root user, not to any user that logs in.

```
chown root.root /home/host1-ftp
```

Anonymous FTP Server Directories

As previously noted, the FTP home directory is named **ftp** and is placed in the **/var** directory. When users log in anonymously, they are placed in this directory. An important part of protecting your system is preventing remote users from using any commands or programs not in the restricted directories. For example, you would not let a user use your **ls** command to list filenames, because **ls** is located in your **/bin** directory. At the same time, you want to let the FTP user list filenames using an **ls** command. Newer FTP daemons such as **vsftpd** and ProFTPD solve this problem by creating secure access to needed system commands and files, while restricting remote users to only the FTP site's directories. In any event, make sure that the FTP home directory is owned by the root user, not by the FTP user. Use the **ls -d** command to check on the ownership of the FTP directory.

```
ls -d /home/ftp
```

To change a directory's ownership, you use the **chown** command, as shown in this example:

```
chown  root.root /home/ftp
```

Another, more traditional solution is to create copies of certain system directories and files needed by remote users and to place them in the **ftp** directory where users can access them. A **bin** directory is placed in the **ftp** directory and remote users are restricted to it, instead of the system's

bin directory. Whenever they use the **ls** command, remote users are using the one in **ftp/bin**, not the one you use in **/bin**. If, for some reason, you set up the anonymous FTP directories yourself, you must use the **chmod** command to change the access permissions for the directories so that remote users cannot access the rest of your system. Create an **ftp** directory and use the **chmod** command with the permission 555 to turn off write access: **chmod 555 ftp**. Next, make a new **bin** directory in the **ftp** directory, and then make a copy of the **ls** command and place it in **ftp/bin**. Do this for any commands you want to make available to FTP users. Then create an **ftp/etc** directory to hold a copy of your **passwd** and **group** files. Again, the idea is to prevent any access to the original files in the **/etc** directory by FTP users. The **ftp/etc/passwd** file should be edited to remove any entries for regular users on your system. All other entries should have their passwords set to **x** to block access. For the **group** file, remove all user groups and set all passwords to **x**. Create an **ftp/lib** directory, and then make copies of the libraries you need to run the commands you placed in the **bin** directory.

Anonymous FTP Files

A directory named **pub**, located in the FTP home directory, usually holds the files you are making available for downloading by remote FTP users. When FTP users log in, they are placed in the FTP home directory (**/home/ftp**), and they can then change to the **pub** directory to start accessing those files (**/home/ftp/pub**). Within the **pub** directory, you can add as many files and directories as you want. You can even designate some directories as upload directories, enabling FTP users to transfer files to your system.

In each subdirectory set up under the **pub** directory to hold FTP files, you should create a **README** file and an **INDEX** file as a courtesy to FTP users. The **README** file contains a brief description of the kind of files held in this directory. The **INDEX** file contains a listing of the files and a description of what each one holds.

Using FTP with rsync

Many FTP servers also support rsync operations using **rsync** as a daemon. This allows intelligent incremental updates of files from an FTP server. You can update multiple files in a directory or a single file such as a large ISO image.

Accessing FTP Sites with rsync

To access the FTP server running an rsync server, you enter the **rsync** command, and following the hostname, you enter a double colon and then either the path of the directory you want to access or one of the FTP server's modules. In the following example, the user updates a local **myproject** directory from the one on the **mytrek.com** FTP site:

```
rsync ftp.mytrek.com::/home/ftp/pub/myproject  /home/myproject
```

To find out what directories are supported by rsync, you check for rsync modules on that site. These are defined by the site's **/etc/rsyncd.conf** configuration file. A *module* is just a directory with all its subdirectories. To find available modules, you enter the FTP site with a double colon only.

```
rsync ftp.mytrek.com::
ftp
```

This tell you that the **ftp.mytrek.com** site has an FTP module. To list the files and directories on the module, you can use the **rsync** command with the **-r** option.

```
rsync -r ftp.mytrek.com::ftp
```

Many sites that run the rsync server will have an rsync protocol that will already be set to access the available rsync module (directory). You can even use **rsync** to update just a single file, such as an ISO image that may have been changed.

Configuring an rsync Server

To configure your FTP server to let clients use rsync on your site, you need to first run rsync as server. Use **services-admin** or **sysv-rc-conf** to turn on the rsync daemon, commonly known as **rsyncd**. This will run the rsync daemon through **xinetd**, using an **rsync** script in **/etc/xinetd.d** to turn it on and set parameters.

When run as a daemon, rsync will read the **/etc/rsyncd.conf** file for its configuration options. Here you can specify FTP options such as the location for the FTP site files. The configuration file is segmented into modules, each with its own options. A module is a symbolic representation of a exported tree (a directory and its subdirectories). The module name is enclosed in brackets, for instance, **[ftp]** for an FTP module. You can then enter options for that module, as by using the path option to specify the location of your FTP site directories and files. The user and group IDs can be specified with the **uid** and **gid** options. The default is nobody. A sample FTP module for anonymous access is shown here:

```
[ftp]
       path = /var/ftp/pub
       comment = ftp site
```

For more restricted access, you can add an **auth users** option to specify authorized users; rsync will allow anonymous access to all users by default. The hosts allow or deny to control access control access from specific hosts. Access to areas on the FTP site by rsync can be further controlled using a secrets file, such as **/etc/rsyncd.secrets**. This is a colon-separated list of user names and passwords.

```
aleina:mypass3
larisa:yourp5
```

A corresponding module to the controlled area would look like this:

```
[specialftp]
      path = /var/projects/special
      command = restricted access
      auth users = aleina,larisa
      secrets file = /etc/rsyncd.secrets
```

If you are on your FTP server and want to see what modules will be made available, you can run **rsync** with the **localhost** option and nothing following the double colon.

```
$ rsync localhost::
ftp
specialftp
```

Remote users can find out what modules you have by entering your hostname and double colon only.

```
rsync ftp.mytrek.com::
```

rsync Mirroring

Some sites will allow you to use rsync to perform mirroring operations. With rsync you do not have to copy the entire site, just those files that have been changed. The following example mirrors the mytrek FTP site to the **/home/ftp/mirror/mytrek** directory on a local system:

```
rsync -a --delete ftp.mytrek.com::ftp /home/ftp/mirror/mytrek
```

The **-a** option is archive mode, which includes several other options, such as **-r** (recursive) to include all subdirectories, **-t** to preserves file times and dates, **-l** recreate symbolic links, and **-p** to preserve all permissions. In addition, the **--delete** option is added to delete files that don't exist on the sending side, removing obsolete files.

The Very Secure FTP Server

The Very Secure FTP Server (vsftpd) is small, fast, easy, and secure. It is designed to avoid the overhead of large FTP server applications like ProFTPD, while maintaining a very high level of security. It can also handle a very large workload, managing high traffic levels on an FTP site. It is perhaps best for sites where many anonymous and guest users will be downloading the same files. Beginning with Red Hat 9, it replaced the Washington University FTP server, WU-FTPD server.

The Very Secure FTP Server is inherently designed to provide as much security as possible, taking full advantage of Unix and Linux operating system features. The server is separated into privileged and unprivileged processes. The unprivileged process receives all FTP requests, interpreting them and then sending them over a socket to the privileged process, which then securely filters all requests. Even the privileged process does not run with full root capabilities, using only those that are necessary to perform its tasks. In addition, the Very Secure FTP Server uses its own version of directory commands like **ls**, instead of the system's versions.

Running vsftpd

The Very Secure FTP Server's daemon is named **vsftpd**. It is designed to be run as a stand-alone server, which can be started and stopped using the **/etc/rc.d/init.d/vsftpd** server script. To start, stop, and restart **vsftpd**, you can use the **service** command.

```
service vsftpd start
```

To have the server start automatically, you can turn it on with the **chkconfig** command and the **on** argument, as shown here. Use the **off** argument to disable the server. If you previously enabled another FTP server such as ProFTPD, be sure to disable it first.

```
chkconfig vsftpd on
```

You can also use **system-config-services** to start and stop **vsftpd**, or to have it started automatically (System | Administration | Services).

Alternatively, you can implement **vsftpd** to be run by **xinetd**, running the server only when a request is made by a user. The use of **xinetd** for the servers is described in detail in Chapter 3. The **xinetd** daemon will run an **xinetd** script file called **vsftpd** located in the **/etc/xinetd.d** directory.

Initially, the server will be turned off. You can turn it on in **xinetd** with the **chkconfig** command and the **on** argument, as shown here. Use the **off** argument to disable the server.

```
chkconfig vsftpd on
```

Restart **xinetd** with the **service** command (or system-config-services) to restart the **vsftpd** server, should you make configuration changes.

```
service xinetd restart
```

Firewall access

To allow firewall access to the FTP port, usually port 21, you should enable access using the firewall configuration tool **system-config-firewall**.

If you are managing your IPtables firewall directly, you could manage access directly by adding the following IPtables rule. This accepts input on port 21 for TCP/IP protocol packages.

```
iptables -A INPUT -p tcp --dport 21 -j ACCEPT
```

Managing vsftpd with system-config-vsftpd

Fedora provides a system administration tool called system-config-vsftpd with which you can configure your vsftpd server easily. First install the **system-config-vsftpd** package. You can then start system-config-vsftpd by choosing System | Administration | Server Settings | FTP Administration. Configuration settings are saved to the **/etc/vsftpd/vsftpd.conf** file.

The system-config-vsftpd tool will display panes for General, Server Control, Access Control, Users, Directory Options, Logging, Network Options, and transfer log (see Figure 21-1). Choose the one you want from the list on the left. The General pane sets basic server features like running ins stand alone mode, the protocol to use, and whether files can be downloaded or uploaded.

The Server Control pane lets you start and stop the server, and examine the server logs.

The Access Control pane lets you deny access to users in the **/etc/vstpd_users.list**. You could specify another file if you wish. You can also ban certain emails, as well as specify the login banner.

The Users pane has two tabs, one for System users and the other for Anonymous users (see Figure 21-2). You can use the anonymous users pane to set controls for anonymous logins, like whether to allow uploading, creating new directories, or not require passwords. You can also specify a different user, site directory, or owner of uploaded files, all to provide greater security for your site.

On the Directory Options pane you can control access to files, hiding files like dot files (configuration files) or any other type of file you may want hidden from anonymous users.

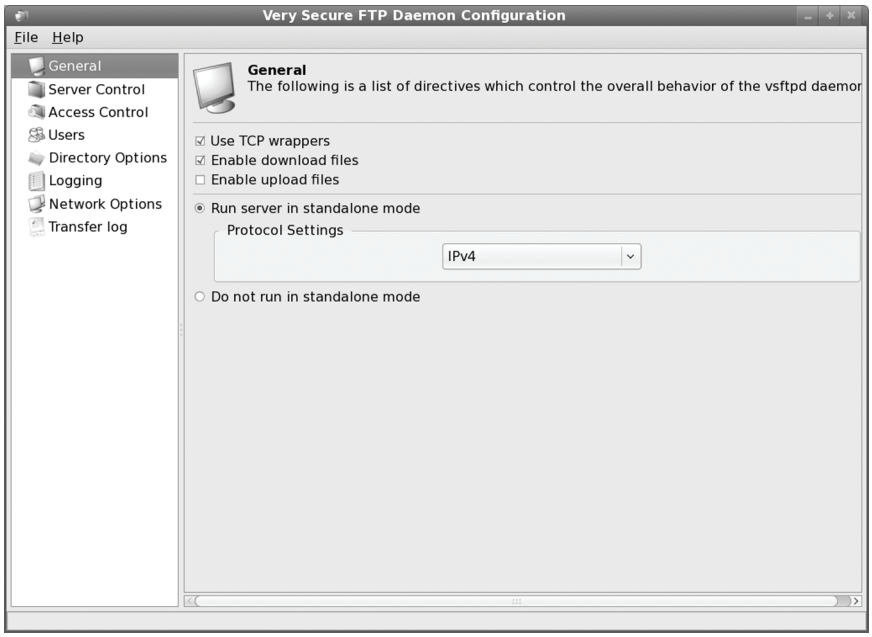


Figure 21-1: system-config-vsftpd General pane

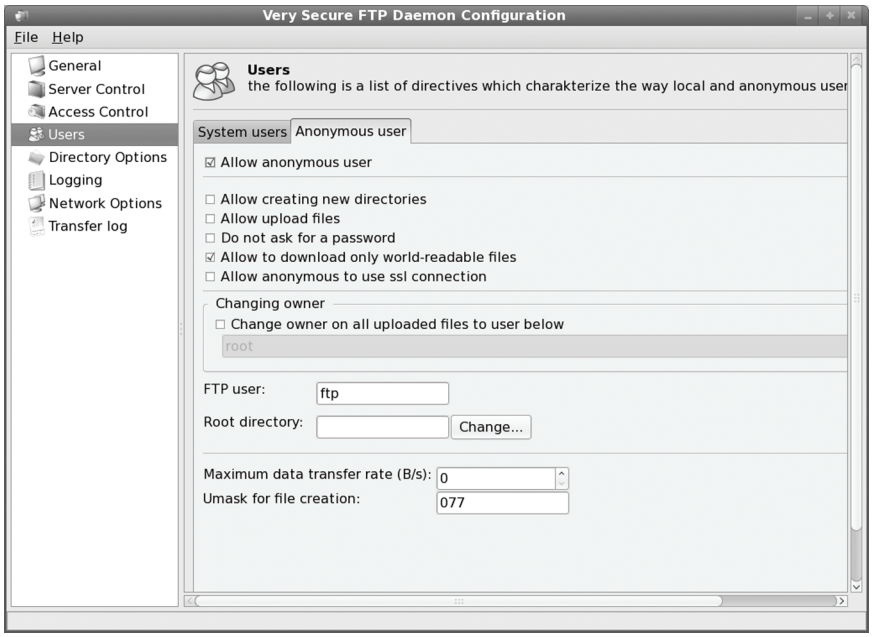


Figure 21-2: system-config-vsftpd Users pane

On the Logging pane you choose logging options as well as specify the logging directory, **/var/log/vsftpd.log** is the default.

On the Network Options pane you can control network connections, setting limits on the number of connections, time-outs for failed connections, and controls for Active and Passive connections (see Figure 21-3).

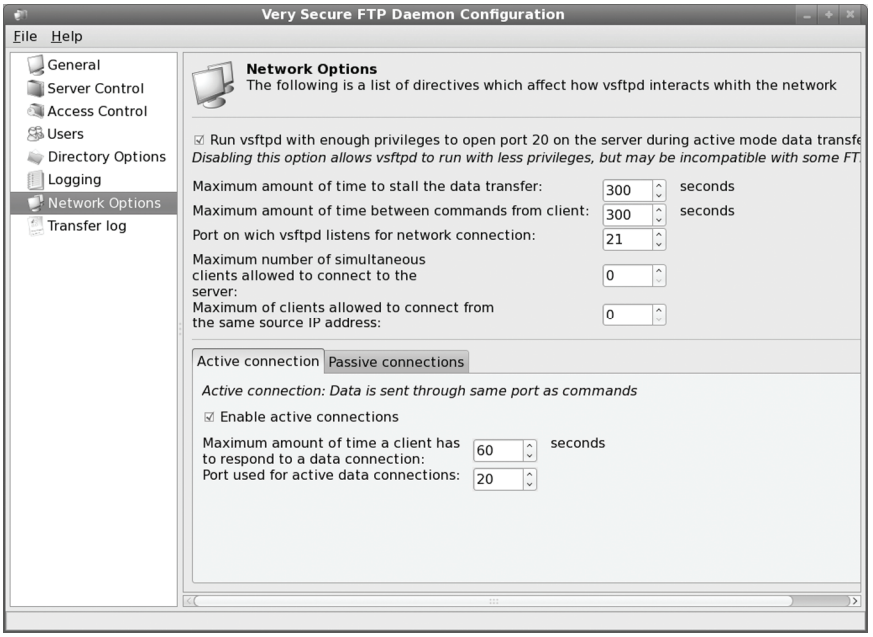


Figure 21-3: system-config-vsftpd Network Options pane

On the Transfer log pane you can display transfer logs for specific dates.

Configuring vsftpd

Configuration for **vsftpd** uses one configuration file, **/etc/vsftpd/vsftpd.conf**. Configuration options are simple and kept to a minimum (see Table 21-2). The **vsftpd.conf** file contains a set of directives where an option is assigned a value (there are no spaces around the = sign). Options can be on and off flags assigned a **YES** or **NO** value, features that take a numeric value, or ones that are assigned a string. A default **vsftpd.conf** file is installed in the **/etc** or **/etc/vsftpd** directory. This file lists some of the commonly used options available with detailed explanations for each. Those that not used are commented out with a preceding # character. Option names are very understandable. For example, **`anon_upload_enable`** allows anonymous users to upload files, whereas **`anon_mkdir_write_enable`** lets anonymous users create directories. The Man page for **vsftpd.conf** lists all options, providing a detailed explanation for each.

Enabling Standalone Access

To run **vsftpd** as a standalone server, you set the listen option to **YES**. This instructs **vsftpd** to continually listen on its assigned port for requests. You can specify the port it listens on with the **listen_port** option.

```
listen=YES
```

Enabling Login Access

In the following example taken from the **vsftpd.conf** file, anonymous FTP is enabled by assigning the **YES** value to the **anonymous_enable** option. The **local_enable** option allows local users on your system to use the FTP server.

```
# Allow anonymous FTP?
anonymous_enable=YES
#
# Uncomment this to allow local users to log in.
local_enable=YES
```

Should you want to let anonymous users log in without providing a password, you can set **no_anon_password** to YES.

Local User Permissions

A variety of user permissions control how local users can access files on the server. If you want to allow local users to create, rename, and delete files and directories on their account, you have to enable write access with the **write_enable** option. This way, any files they upload, they can also delete. Literally, the **write_enable** option activates a range of commands for changing the file system, including creating, renaming, and deleting both files and directories. With **user_config_dir** you can configure specific users.

```
write_enable=YES
```

You can further specify the permissions for uploaded files using the **local_umask** option (022 is the default set in **vsftpd.conf**, read and write for the owner and read-only for all other users, 644).

```
local_umask=022
```

Because ASCII uploads entail certain security risks, they are turned off by default. However, if you are uploading large text files, you may want to enable them in special cases. Use **ascii_upload_enable** to allow ASCII uploads.

Anonymous User Permissions

You can also allow anonymous users to upload and delete files, as well as create or remove directories. Uploading by anonymous users is enabled with the **anon_upload_enable** option. To let anonymous users also rename or delete their files, you set the **anon_other_write_enable** option. To let them create directories, you set the **anon_mkdir_write_enable** option.

```
anon_upload_enable=YES
anon_other_write_enable=YES
anon_mkdir_write_enable=YES
```

Option	Description
`listen`	Set standalone mode.
`listen_port`	Specify port for standalone mode.
`anonymous_enable`	Enable anonymous user access.
`local_enable`	Enable access by local users.
`no_anon_password`	Specify whether anonymous users must submit a password.
`anon_upload_enable`	Enable uploading by anonymous users.
`anon_mkdir_write_enable`	Allow anonymous users to create directories.
`aonon_world_readable_only`	Make uploaded files read-only to all users.
`idle_session_timeout`	Set time limit in seconds for idle sessions.
`data_connection_timeouts`	Set time limit in seconds for failed connections.
`dirmessage_enable`	Display directory messages.
`ftpd_banner`	Display FTP login message.
`xferlog_enable`	Enable logging of transmission transactions.
`xferlog_file`	Specify log file.
`deny_email_enable`	Enable denying anonymous users, whose e-mail addresses are specified in **vsftpd.banned**.
`userlist_enable`	Deny access to users specified in the **vsftp.user_list** file.
`userlist_file`	Deny or allow users access depending on setting of `userlist_deny`.
`userlist_deny`	When set to **YES**, **userlist_file** deny list users access. When set to **NO**, **userlist_file** allow list users, and only those users, access.
`chroot_list_enable`	Restrict users to their home directories.
`chroot_list_file`	Allow users access to home directories. Unless `chroot_local_user` is set to **YES**, this file contains a list of users not allowed access to their home directories.
`chroot_local_user`	Allow access by all users to their home directories.
`pam_service_name`	Specify PAM script.
`ls_recurse_enable`	Enable recursive listing.
`user_config_dir`	Directory for user specific configurability

Table 21-2: Configuration Options for vsftpd.conf

The `anon_world_readable_only` option will make uploaded files read-only (downloadable), restricting write access to the user that created them. Only the user that uploaded a file can delete it.

All uploaded files are owned by the anonymous FTP user. You can have the files owned by another user, adding greater possible security. In effect, the actual user owning the uploaded files becomes hidden from anonymous users. To enable this option, you use **chown_uploads** and specify the new user with **chown_username**. Never make the user an administrative user like **root**.

```
chown_uploads=YES
chown_username=myftpfiles
```

The upload directory itself should be given write permission by other users.

```
chmod 777 /var/ftp/upload
```

You can control the kind of access that users have to files with the **anon_mask** option, setting default read/write permissions for uploaded files. The default is 077, which gives read/write permission to the owner only (600). To allow all users read access, you set the umask to 022, where the 2 turns off write permission but sets read permission (644). The value 000 allows both read and write for all users.

Connection Time Limits

To more efficiently control the workload on a server, you can set time limits on idle users and failed transmissions. The **idle_session_timeout** option will cut off idle users after a specified time, and **data_connection_timeouts** will cut off failed data connections. The defaults are shown here:

```
idle_session_timeout=600
data_connection_timeout=120
```

Messages

The **dirmessage_enable** option allows a message held in a directory's **.message** file to be displayed whenever a user accesses that directory. The **ftpd_banner** option lets you set up your own FTP login message. The default is shown here:

```
ftpd_banner=Welcome to blah FTP service.
```

Logging

A set of **xferlog** options control logging. You can enable logging, as well as specify the format and the location of the file.

```
xferlog_enable=YES
```

Use **xferlog_file** option to specify the log file you want to use. The default is shown here:

```
xferlog_file=/var/log/vsftpd.log
```

vsftpd Access Controls

Certain options control access to the FTP site. As previously noted, the **anonymous_enable** option allows anonymous users access, and **local_enable** permits local users to log in to their accounts (If **/etc/vsftpd** exits, no **vsftpd.** prefix is used on files).

Denying Access

The **deny_email_enable** option lets you deny access by anonymous users, and the **banned_email** file option designates the file (usually **vstfpd.banned**) that holds the e-mail addresses of those users. The **vsftpd.ftpusers** file lists those users that can never be accessed. These are usually system users like **root**, **mail**, and **nobody**. See Table 21-3 for a list of vsftpd files.

File	Description
vsftpd.ftpusers	Users always denied access
vsftpd.user_list	Specified users denied access (allowed access if **userlist_deny** is **NO**)
vsftpd.chroot_list	Local users allowed access (denied access if **chroot_local_user** is on)
/etc/vsftpd.conf	vsftpd configuration file
/etc/pam.d/vsftpd	PAM vsftpd script
/etc//init.d/vsftpd	Service vsftpd server script, standalone
/etc/xinetd.d/vsftpd	Xinetd vsftpd server script

Table 21-3: Files for vsftpd

User Access

The **userlist_enable** option controls access by users, denying access to those listed in the file designated by the **userlist_file** option (usually **vsftpd.user_list**). If, instead, you want to restrict access to just certain select users, you can change the meaning and usage of the **vsftpd.user_list** file to indicate only those users allowed access, instead of those denied access. To do this, you set the **userlist_deny** option to **NO** (its default is **YES**). Only users listed in the **vsftpd.user_list** file will be granted access to the FTP site.

User Restrictions

The **chroot_list_enable** option controls access by local users, letting them access only their home directories, while restricting system access. The **chroot_list_file** option designates the file (usually **vstfpd.chroot**) that lists those users allowed access. You can allow access by all local users with the **chroot_local_user** option.

```
chroot_local_users=YES
```

If this option is set, then the file designated by **chroot_list_file** will have an inverse meaning, listing those users not allowed access. In the following example, access by local users is limited to those listed in **vsftpd.chroot**:

```
chroot_list_enable=YES
chroot_list_file=/etc/vsftpd.chroot_list
```

On Fedora the **secure_chroot_dir** option is used to specify a non-user secure root directory.

```
secure_chroot_dir=/var/run/vsftpd
```

User Authentication

The **vsftpd** server makes use of the PAM service to authenticate local users that are remotely accessing their accounts through FTP. In the **vsftpd.conf** file, the PAM script used for the server is specified with the **pam_service_name** option.

```
pam_service_name=vsftpd
```

In the **etc/pam.d** directory, you will find a PAM file named **vsftpd** with entries for controlling access to the **vsftpd** server. PAM is currently set up to authenticate users with valid accounts, as well as deny access to users in the **/etc/vsftpd.ftpusers** file. The default **/etc/pam.d/vsftpd** file is shown here:

```
#%PAM-1.0
auth required pam_listfile.so item=user sense=deny
                file=/etc/vsftpd.ftpusers onerr=succeed
auth    required  pam_stack.so service=system-auth
auth   required   pam_shells.so
account required  pam_stack.so service=system-auth
session required  pam_stack.so service=system-auth
```

Command Access

Command usage is highly restricted by vsftpd. Most options for the **ls** command that lists files are not allowed. Only the asterisk file-matching operation is supported. To enable recursive listing of files in subdirectories, you have to enable the use of the **-R** option by setting the **ls_recurse_enable** option to **YES**. Some clients, such as **ncftp**, will assume that the recursive option is enabled.

vsftpd Virtual Hosts

Though the capability is not inherently built in to vsftpd, you can configure and set up the vsftpd server to support virtual hosts. *Virtual hosting* is where a single FTP server operates as if it has two or more IP addresses. Several IP addresses can then be used to access the same server. The server will then use a separate FTP user directory and files for each host. With vsftpd, this involves manually creating separate FTP users and directories for each virtual host, along with separate vsftpd configuration files for each virtual host in the **/etc/vsftpd** directory. **vsftpd** is configured to run as a standalone service. Its **/etc/init.d/vsftpd** startup script will automatically search for and read any configuration files listed in the **/etc/vsftpd** directory.

If, on the other hand, you wish to run **vsftpd** as a **xinetd** service, you have to create a separate **xinetd** service script for each host in the **/etc/xinetd.d** directory. In effect, you have several **vsftpd** services running in parallel for each separate virtual host. The following example uses two IP addresses for an FTP server:

1. Create an FTP user for each host. Create directories for each host (you can use the one already set up for one of the users). For example, for the first virtual host you could use **FTP-host1**. Be sure to set root ownership and the appropriate permissions.

```
useradd -d /home/ftp-host1 FTP-host1
chown root.root /var/ftp-host1
chmod a+rx /home/ftp-host1
```

```
umask 022
mkdir /home/ftp-host1/pub
```

2. Set up two corresponding vsftpd service scripts in the **/etc/xinetd.d** directory. The **vsftpd** directory in **/usr/share/doc** has an **xinetd** example script, **vsftpd.xinetd**. Within each, enter a **bind** command to specify the IP address the server will respond to.

```
bind  192.168.0.34
```

3. Within the same scripts, enter a **server_args** entry specifying the name of the configuration file to use.

```
server_args = vsftpd-host1.conf
```

4. Within the **/etc/vsftpd** directory, create separate configuration files for each virtual host. Within each, specify the FTP user you created for each, using the **ftp_username** entry.

```
ftp_username = FTP-host1
```

vsftpd Virtual Users

Virtual users can be implemented by making use of PAM to authenticate authorized users. In effect, you are allowing access to certain users, while not having to actually set up accounts for them on the FTP server system. First, create a PAM login database file to use along with a PAM file in the **/etc/pam.d** directory that will access the database. Then create a virtual FTP user along with corresponding directories that the virtual users will access (see the vsftpd documentation at **http://vsftpd.beasts.org** for more detailed information). Then, in the **vsftpd.conf** file, you can disable anonymous FTP:

```
anonymous_enable=NO
local_enable=YES
```

and then enable guest access:

```
guest_enable=YES
guest_username=virtual
```

Note: ProFTPD is based on the same design as the Apache web server, implementing a similar simplified configuration structure and supporting such flexible features as virtual hosting. ProFTPD is an open source project made available under a GPL license. You can download the current version from its website at **www.proftpd.org**. There you will also find detailed documentation including FAQs, user manuals, and sample configurations. Check the site for new releases and updates.

fedora

22. Web Servers

The primary web server for Fedora is Apache, which has almost become the standard web server for all Linux distributions. It is a very powerful, stable, and fairly easy-to-configure system. Fedora provide default configuration for the Apache, making it usable as soon as they are installed. Also available is the **lighttpd**, a small fast Web server with few system requirements, and both **cherokee** and thttpd, tiny stripped down very fast Web servers. Table 22-1 lists various Web server Web sites.

In additions, the Zope Application server for Web development is included on the Fedora repository. Zope application server (**www.zope.org**) is an open source web server with integrated security, web-based administration and development, and database interface features. It was developed by the Zope Corporation, which also developed the Python programming language.

Website	Description
www.apache.org	Apache Software Foundation
http://httpd.apache.org	Apache HTTP Server Project
http://jakarta.apache.org	Jakarta Apache Project
www.lighthttpd.net	The Lighthttpd Web server
www.cherokee-project.com	The cherokee Web server
www.lighthttpd.net	The lighthttpd server
www.acme.com/software/thttpd/	The thttpd tiny http server
www.php.net	PHP Hypertext Preprocessor, embedded web page programming language
www.zope.org	Zope application server

Table 22-1: Web server Websites

Lighttpd Web Server

The Lighttpd Web server is a very fast light-weight Web server. The Lighttpd Web site is **http://www.lighttpd.net**. On the site you can access documentation which includes tutorials and detailed descriptions.

Lighttpd uses the **lighttpd** directory in the Web server site directory, **/var/www/lighttpd**. Its configuration files are located at **/etc/lighttpd**. Server variable sare set in the **/etc/sysconfig/lighttpd** file. You can start and stop the server using the **/etc/init.d/lighttpd** service script which is also manages with **system-config-services** as **lighttpd** (System | Administration | Services).

```
service lighttpd start
```

Just as with Apache, you can access the Web server using the localhost URL with port 80.

```
http://localhost:80
```

The Lighttpd configuration file is **/etc/lighttpd/lighttpd.conf**. The configuration file holds a series of simple Web configuration options such as **server.document-root** for the Web site files **(/var/www/lighttpd** is the default), **server.errorlog** for the log file, and **index-file.names** for he

name of the index HTML file to use. The sever user name (**server.username**) is set to lighttpd. Entries can also be conditionals, used to check for certain values like a virtual Web address, and then, within an attached block, list options or conditionals that apply if the condition is true.

Detailed documentation is located in the **/usr/share/doc/lighttpd*** directory, which holds text files for different modules and features, like **evhost.txt** for Enhanced virtual hosts and **authentication.txt** for authentication methods. Check the **configuration.txt** file for a detailed description of configuration options with examples.

Lighttpd can be expanded easily using supported modules. Several are included such as mod_simple-vhost for virtual hosts, though almost all are initially disabled in the **lighttpd.conf** file. In the **server.modules** block at the top of the file, you will file the basic modules listed, though commented out. The module name begins with a **mod_** prefix, as in **mod_auth**. Remove the comment character (#) on a module's entry to enable loading of the module. The Fedora repository also provides packages for additional modules including FastCGI, GeoIP for locations, and virtual hosts with MySQL database support. The Lighttpd modules are located in the **/usr/lib/lighttpd** directory.

Should you want to run both Apache and Lighttpd, you could change the port that Lighttpd uses to 81 in the **/etc/lighttpd/lighttphd.conf** file.

```
server.port = 81
```

Simple virtual hosts can be set up with the **simple-vhost** options for server, default-host, and document-root (**mod_simple_vhost** module). For enhanced virtual hosts, Lighttpd will rewrite the URL to a predetermined directory specified by the **evhost.path-pattern** option (**mod_evhost** module).

Apache Web Server

The Apache web server is a full-featured free HTTP (web) server developed and maintained by the Apache Server Project. The aim of the project is to provide a reliable, efficient, and easily extensible web server, with free open source code made available under its own Apache Software License. The server software includes the server daemon, configuration files, management tools, and documentation. The Apache Server Project is maintained by a core group of volunteer programmers and supported by a great many contributors worldwide. The Apache Server Project is one of several projects currently supported by the Apache Software Foundation (formerly known as the Apache Group). This nonprofit organization provides financial, legal, and organizational support for various Apache Open Source software projects, including the Apache HTTPD Server, Java Apache, Jakarta, and XML-Apache. The website for the Apache Software Foundation is **www.apache.org**. Table 22-1 lists various Apache-related websites.

Apache was originally based on the NCSA web server developed at the National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign. Apache has since emerged as a server in its own right and become one of the most popular web servers in use. Although originally developed for Linux and Unix systems, Apache has become a cross-platform application with Windows and OS/2 versions. Apache provides online support and documentation for its web server at **http://httpd.apache.org**. An HTML-based manual is also provided with the server installation.

Java: Apache Jakarta Project

The Apache Jakarta Project supports the development of Open Source Java software; its website is located at **http://jakarta.apache.org**. Currently, the Jakarta supports numerous projects, including libraries, tools, frameworks, engines, and server applications. Tomcat is an open source implementation of the Java Servlet and JavaServer Pages specifications. Tomcat is designed for use in Apache servers. JMeter is a Java desktop tool to test performance of server resources, such as server lets and CGI scripts. Velocity is a template engine that provides easy access to Java objects. Watchdog is a tool that checks the compatibility of servlet containers. Struts, Cactus, and Tapestry are Java frameworks, established methods for developing Java web applications.

Tip: Check the Fedora Project JAVA FAQ for more information about Java, **www.fedoraproject.org/wiki/JavaFAQ**.

Web Site Directories	Description
/var/www/html	Web site Web files
/var/www/cgi-bin	CGI program files
/var/www/html/manual	Apache Web server manual
Configuration Files	
.htaccess	Directory-based configuration files; an **.htaccess** file holds directives to control access to files within the directory in which it is located
/etc/httpd/conf	Directory for Apache Web server configuration files
/etc/httpd/conf/httpd.conf	Apache Web server configuration file
/etc/httpd/conf.d	Directory holding module configuration files like **ssl.conf** for SSL, **php.conf** for PHP, and welcome.conf for the Welcome message.
Service Scripts	
/etc/rc.d/init.d/httpd	Service script for Web server daemon
/etc/sysconfig/httpd	Red Hat and Fedora configuration options for Web server daemon, as used by the httpd service script
Application and Module Files	
/usr/sbin	Location of the Apache Web server program file and utilities
/usr/share/doc/	Apache Web server documentation
/var/log/http	Location of Apache log files
/etc/httpd/modules	Directory holding Apache modules, link to /usr/lib/httpd/modules
/etc/httpd/run	Directory holding Apache process IDs, link to /var/run

Table 22-2: Apache Web Server Files and Directories

Linux Apache Installation

Your Linux distribution will normally provide you with the option of installing the Apache Web server during your initial installation of your Linux system. All the necessary directories and configuration files are automatically generated for you. Then, whenever you run Linux, your system is already a fully functional Web site. Every time you start your system, the Web server will also start up, running continuously. On most distributions, the directory reserved for your Web site data files is **/var/www/html**. Place your Web pages in this directory or in any subdirectories. Your system is already configured to operate as a Web server. All you need to do is perform any needed network server configurations, and then designate the files and directories open to remote users. You needn't do anything else. Once your Web site is connected to a network, remote users can access it.

The Web server normally sets up your Web site in the **/var/www** directory. It also sets up several directories for managing the site. The **/var/www/cgi-bin** directory holds the CGI scripts, and **/var/www/html/manual** holds the Apache manual in HTML format. You can use your browser to examine it. Your Web pages are to be placed in the **/var/www/html** directory. Place your Web site home page there. Your configuration files are located in a different directory, **/etc/httpd/conf**. Table 22-2 lists the various Apache Web server directories and configuration files.

Apache also installs several management applications such as **apachectl** for starting and stopping the server. The **/etc/init.d/httpd** script is designed to safely run **apachectl**, and should be used to actually start and stop the server manually. It is also used by **system-config-services** to start and stop the Web server when your system starts up an shuts down. You can use system-config-httpd to configure your Web server, accessible from System | Administration | Server Settings | HTTP. Table 22-3 lists some of the applications. See http://httpd.apache.org/docs/2.0/programs/ for a complete list.

Application	Description
apachectl	Control start, stop, and restart the apache server
system-config-httpd	Fedora desktop configuration tool for the Apache Web server
htcacheclean	Limit the size of the Web site disk cache
ab	Tool to benchmark Web site performance
htpasswd	Manage passwords for Web site directories
/etc/init.d/httpd	Script designed for Fedora to start, stop, and restart server, invoked apachectl. Managed by **system-config-services**, System | Administration | Services.

Table 22-3: Apache management tools

Apache Multiprocessing Modules: MPM

Apache now use a new architecture that uses multiprocessing modules (MPMs), which are designed to customize Apache to different operating systems, as well as handle certain multiprocessing operations. For the main MPM, a Linux system would use either the prefork or worker MPM, whereas Windows would use the mpm_winnt MPM. The prefork is a standard MPM module designed to be compatible for older Unix and Linux systems, particularly those that do not support threading. It is the module loaded by default by Fedora. The worker MPM implements

threading for Apache server processes, a feature supported by the Native POSIX Thread Libraries (NPTL) that are part of Red Hat and Fedora. You can configure the workload parameters for both in the Apache configuration file, **/etc/httpd/conf/httpd.conf**.

Many directives that once resided in the Apache core are now placed in respective modules and MPMs. With this modular design, several directives have been dropped, such as ServerType. Configuration files for these module are located in the **/etc/httpd/conf.d** directory.

Starting and Stopping the Web Server

On most systems, Apache is installed as a stand-alone server, continually running. As noted in Chapter 3, in the discussion of init scripts, your system automatically starts up the Web server daemon, invoking it whenever you start your system. A service script for the Web server called **httpd** is in the **/etc/rc.d/init.d** directory. Symbolic links through which this script is run are located in corresponding runlevel directories. You will usually find the **S85httpd** link to **/etc/rc.d/init.d/httpd** in the runlevel 3 and 5 directories, **/etc/rc.d/rc3.d** and **/etc/rc.d/rc5.d**. You can use the **chkconfig** command or the System V Init Editor to set the runlevels at which the httpd server will start, creating links in appropriate runlevel directories. The following command will set up the Web server (httpd) to start up at runlevels 3 and 5.

```
chkconfig --level 35 httpd on
```

You can also use **service** command to start and stop the httpd server manually. This may be helpful when you are testing or modifying your server. The **httpd** script with the **start** option starts the server, the **stop** option stops it, and **restart** will restart it. Simply killing the Web process directly is not advisable.

```
/sbin/service httpd restart
```

You can also run the httpd script directly.

```
service  httpd  start
```

The **/etc/sysconfig/httpd** configuration file is used to set options for the **httpd** service script. Here you can specify such options as whether to use the worker or prefork Apache MPM modules.

Apache also provides a control tool called **apachectl** (Apache control) for managing your Web server. With **apachectl**, you can start, stop, and restart the server from the command line. The **apachectl** command takes several arguments: **start** to start the server, **stop** to stop it, restart to shut down and restart the server, and **graceful** to shut down and restart gracefully. In addition, you can use **apachectl** to check the syntax of your configuration files with the **config** argument. You can also use **apachectl** as a system service file for your server in the **/etc/rc.d** directory.

Remember, **httpd** is a script that calls the actual **httpd** daemon. You could call the daemon directly using its full pathname. This daemon has several options. The **-d** option enables you to specify a directory for the httpd program if it is different from the default directory. With the **-f** option, you can specify a configuration file different from **httpd.conf**. The **-v** option displays the version.

```
/usr/sbin/httpd -v
```

To check your Web server, start your Web browser and enter the Internet domain name address of your system. For the system **turtle.mytrek.com,** the user enters `http://turtle.mytrek.com`. This should display the home page you placed in your Web root directory. A simple way to do this is to use Lynx, the command line Web browser. Start Lynx**,** and type **g** to open a line where you can enter a URL for your own system. Lynx displays your Web site's home page. Be sure to place an **index.html** file in the **/var/www/html** directory first.

Once you have your server running, you can check its performance with the **ab** benchmarking tool, also provided by Apache: **ab** shows you how many requests at a time your server can handle. Options include **-v,** which enables you to control the level of detail displayed, **-n,** which specifies the number of requests to handle (default is 1), and **-t,** which specifies a time limit.

Note: Currently there is no support for running Apache under **xinetd**. In Apache 2.0, such support is determined by choosing an MPM module designed to run on **xinetd**.

Apache Configuration

Configuration directives are placed in the **httpd.conf** configuration file. A documented version of the **httpd.conf** configuration file is installed automatically in **/etc/httpd/conf**. It is strongly recommended that you consult this file on your system. It contains detailed descriptions and default entries for Apache directives.

Any of the directives in the main configuration files can be overridden on a per-directory basis using an **.htaccess** file located within a directory. Although originally designed only for access directives, the **.htaccess** file can also hold any resource directives, enabling you to tailor how Web pages are displayed in a particular directory. You can configure access to .**htaccess** files in the **httpd.conf** file.

In addition, many of the modules provided for Apache have their own configuration files. These are places in the **/etc/httpd/conf.d** directory.

Apache Configuration with system-config-httpd

You can use the Fedora Web administration tool, system-config-httpd, to perform basic configuration tasks, using a desktop interface. Choose System | Administration | Server Settings | httpd. This opens the HTTP Server Configuration window with four tabs: Main, Virtual Hosts, Server, and Performance Tuning (see Figure 22-1). In each of these you will see buttons to open dialog boxes where you can enter settings. Click the Help button to display a Web page–based reference manual that details how to use each tab.

- On the Main tab, you enter your Web server address, the Web master's e-mail address, and the addresses and ports the Web server will be listening on.
- On the Virtual Hosts tab, be sure to select Default Virtual Host and click Edit to set the default settings for server options, pages searches, SSL support, log files, CGI environment support, and directories (Performance). To add a virtual host, click Add to open a window where you can enter host information such as the virtual hostname and IP address. You can select different configuration tabs for the virtual host, such as log files and directory controls.

- On the server tab, you set administrative settings such as the Apache server's user ID and the process ID file, along with the user and group.
- The Performance Tuning tab lets you set different usage limits such as the maximum number of requests and the number of requests per connection.

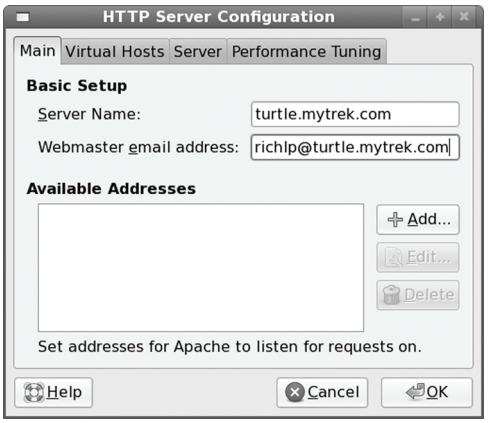


Figure 22-1: system-config-httpd Server Configuration

When the Apache Configuration Tool saves its settings, it will overwrite the Apache configuration file, **/etc/httpd/conf/httpd.conf**. It is advisable that you first make a backup copy of your httpd.conf file in case you want to restore the original settings created by your distribution for Apache. If you have already manually edited this file, you will receive a warning, and the Apache Configuration Tool will make a backup copy in **/etc/httpd/conf/httpd.conf.bak**.

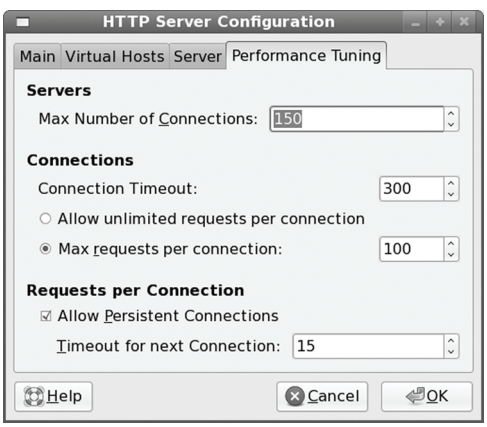


Figure 22-2: system-config-httpd Performance Tuning

The Server tab sets server features like the server's user and group. The Performance Tuning tab sets limits on connections, like the maximum number of connections and the period for time-out (see Figure 22-2)

The Virtual Hosts pane is where you can configure particular sites, including the default site (see Figure 22-3). Virtual hosts are listed, beginning with the Default Virtual Host. Use the Add button to add a new host, and the Edit button to modify current ones. The delete button removes the host. To change global settings, click the Edit Default Settings button.

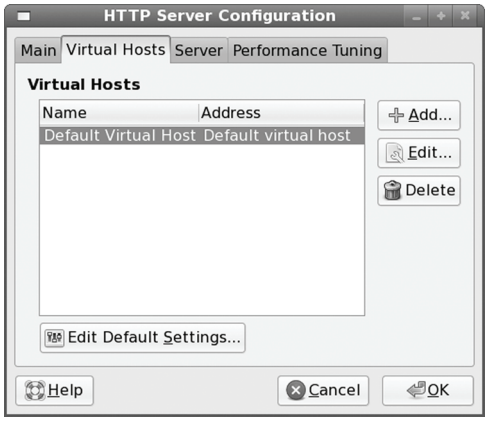


Figure 22-3: system-config-httpd Virtual Hosts

Adding or editing a virtual host opens the Virtual Hosts Properties window, listing five tabs: General Options, Page Options, SSL, Logging, Environment, and Performance (see Figure 22-4).

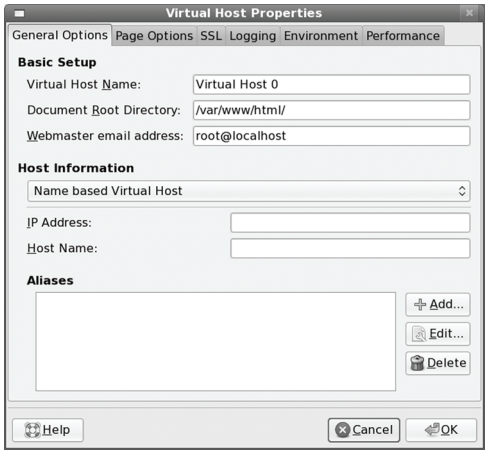


Figure 22-4: system-config-httpd Virtual Hosts Properties, General Options

On the General Options tab you specify the host name, is site directory, administrator address and the type of virtual host, Name or IP. You can also add aliases.

On Page Options you can specify the type of file that will automatically be searched for and displayed. This is usually index.html and any of its variations like **index.htm**. You could specify your own.

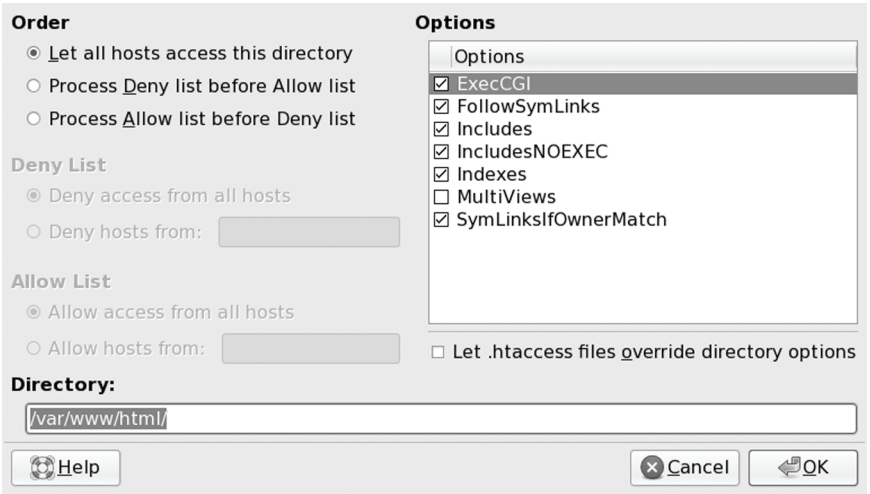


Figure 22-5: system-config-httpd Virtual Hosts Properties, Performance, Directories

On the SSL tab you can enable SSL support, specifying certificates to use along with SSL options.

The Logging tab specifies the log and error logs.

On the Environment tab you can define environment variables for your CGI scripts, as well as those you want to pass to your scripts and those you want undefined.

The Performance tab control directory access features like access controls and directory capabilities like executing CGI scripts or following symbolic links. At the top of the tab is a button for editing default directory options. This opens a window that lists possible directory options, with the enabled ones checked.

From the directory list you can select a particular directory to edit. Click the Add button to add a new directory. This opens a Directory Options window with the possible options displayed on the right and the access control options on the left (see Figure 22-5). For Deny and Allow lists you can specify your own files.

Apache Configuration Directives

Apache configuration operations take the form of directives entered into the Apache configuration files. With these directives, you can enter basic configuration information, such as your server name, or perform more complex operations, such as implementing virtual hosts. The design is flexible enough to enable you to define configuration features for particular directories and different virtual hosts. Apache has a variety of different directives performing operations as diverse as controlling directory access, assigning file icon formats, and creating log files. Most directives set values such as **`DirectoryRoot`**, which holds the root directory for the server's web pages, or **`Port`**, which holds the port on the system that the server listens on for requests. The syntax for a simple directive is shown here:

```
directive option option ...
```

Certain directives create blocks able to hold directives that apply to specific server components (also referred to as sectional directives). For example, the **Directory** directive is used to define a block within which you place directives that apply only to a particular directory. Block directives are entered in pairs: a beginning directive and a terminating directive. The terminating directive defines the end of the block and consists of the same name beginning with a slash. Block directives take an argument that specifies the particular object to which the directives apply. For the **Directory** block directive, you must specify a directory name to which it will apply. The **<Directory** *mydir***>** block directive creates a block whose directives within it apply to the *mydir* directory. The block is terminated by a **</Directory>** directive. The **<VirtualHost** *hostaddress***>** block directive is used to configure a specific virtual web server and must include the IP or domain name address used for that server. **</VirtualHost>** is its terminating directive. Any directives you place within this block are applied to that virtual web server. The **<Limit** *method***>** directive specifies the kind of access method you want to limit, such as GET or POST. The access control directives located within the block list the controls you are placing on those methods. The syntax for a block directive is as follows:

```
<block-directive option ... >
 directive option ...
 directive option ...
</block-directive>
```

Global directives are placed in one of the main configuration files. Directives for particular sites are located in that sites configuration file in **/etc/httpd/sites-available** directory. Directory directives in those files can be used to configure a particular directory. However, Apache also makes use of directory-based configuration files. Any directory may have its own **.htaccess** file that holds directives to configure only that directory. If your site has many directories, or if any directories have special configuration needs, you can place their configuration directives in their **.htaccess** files, instead of filling the main configuration file with specific **Directory** directives for each one. You can control what directives in an **.htaccess** file take precedence over those in the main configuration files. If your site allows user- or client-controlled directories, you may want to carefully monitor or disable the use of **.htaccess** files in them. (It is possible for directives in an **.htaccess** file to override those in the standard configuration files unless disabled with AllowOverride directives.)

You can find a complete listing of Apache web configuration directives at the Apache website, **http://httpd.apache.org/docs/2.2/mod/directives.html**.

Global Configuration

The standard Apache configuration has three sections: global settings, server settings, and virtual hosts. The global settings control the basic operation and performance of the web server. Here you set configuration locations, process ID files, timing, settings for the MPM module used, and what Apache modules to load.

The **ServerTokens** directive prevents disclosure of any optional modules your server is using. The **ServerRoot** directive specifies where your web server configuration files and modules are kept. This server root directory is then used as a prefix to other directory entries.

```
ServerRoot /etc/httpd
```

On Fedora the Process ID file is in the **/etc/httpd/run/httpd.pid** file. The **/etc/httpd/run** directory holds several PID file for various Apache supported processes.

Connection and request timing is handled by **Timeout**, **KeepAlive**, **MaxKeepAlive**, and **KeepAliveTimeout** directives. **Timeout** is the time in seconds that the web server times out a send or receive request. **KeepAlive** allows persistent connections, several requests from a client on the same connection. This is turned off by default. **KeepAliveRequests** sets the maximum number of requests on a persistent connection. **KeepAliveTimeout** is the time that a given connection to a client is kept open to receive more requests from that client.

The **Listen** directive will bind the server to a specific port or IP address. By default this is port 80. The Listen directive is defined in **httpd.conf**.

```
Listen 80
```

MPM Configuration

Configuration settings for MPM prefork and worker modules let you tailor your Apache web server to your workload demands. Default entries will already be set for a standard web server operating under a light load. You can modify these settings for different demands.

Two MPM modules commonly available to Unix and Linux systems are prefork and worker. The prefork module supports one thread per process, which maintains compatibility with older systems and modules. The worker module supports multiple threads for each process, placing a much lower load on system resources. They share several of the same directives, such as **StartServer** and **MaxRequestPerChild**. Fedora currently uses the worker modules.

Apache runs a single parent process with as many child processes as are needed to handle requests. Configuration for MPM modules focuses on the number of processes that should be available. The prefork module will list server numbers, as a process is started for each server; the worker module will control threads, since it uses threads for each process. The **StartServer** directives lists the number of server processes to start for both modules. This will normally be larger for the prefork than for the worker module.

In the prefork module you need to set minimum and maximum settings for spare servers. **MaxClients** sets the maximum number of servers that can be started, and **ServerLimit** sets the number of servers allowed. The **MaxRequestsPerChild** sets the maximum number of requests allowed for a server.

In the worker module, **MaxClients** also sets the maximum number of client threads, and **ThreadsPerChild** sets the number of threads for each server. **MaxRequestsPerChild** limits the maximum number of requests for a server. Spare thread limits are also configured.

The directives serve as a kind of throttle on the web server access, controlling processes to keep available and limit the resources that can be used. In the prefork configuration, the **StartServer** is set number to 8, and the spare minimum to 5, with the maximum spare as 20. This means that initially 8 server processes will be started up and will wait for requests, along with 5 spare processes. When server processes are no longer being used, they will be terminated until the number of these spare processes is less than 20. The maximum number of server processes that can be started is 256. The maximum number of connections per server process is set at 4,000.

In the worker MPM, only 2 server processes are initially started. Spare threads are set at 25 and 75. The maximum number of threads is set at 150, with the threads per child at 25.

Server Configuration

Certain directives are used to configure your server's overall operations. These directives are placed midway in the **httpd.conf** configuration file, directly under the section labeled Server Settings. Some directives require pathnames, whereas others only need to be turned on or off with the keywords **on** and **off**. The default **httpd.conf** file already contains these directives. Some are commented out with a preceding **#** symbol. You can activate a directive by removing its **#** sign. Many of the entries are preceded by comments explaining their purpose.

The following is an example of the **ServerAdmin** directive used to set the address where users can send mail for administrative issues. You replace the **you@your.address** entry with the address you want to use to receive system administration mail. By default, this is set to **root@localhost**.

```
# ServerAdmin: Your address, where problems should be e-mailed.
ServerAdmin you@your.address
```

A Web server usually uses port 80, which is the Apache default. If you want to use a different port, specify it with the **Port** directive.

```
Port 80
```

The **ServerName** directive holds the hostname for your Web server. Specifying a hostname is important to avoid unnecessary DNS lookup failures that can hang your server. Notice the entry is commented with a preceding **#**. Simply remove the **#** and type your Web server's hostname in place of *new.host.name*. If you are using a different port than 80, be sure to specify it attached to the host name, as in **turtle.mytrek.com:80**. Here is the original default entry:

```
# ServerName allows you to set a hostname which is sent
# back to clients for your server if it's different than the
# one the program would get (i.e. use
# "www" instead of the host's real name).

#ServerName new.host.name:80
```

A modified **ServerName** entry would look like this:

```
ServerName turtle.mytrek.com
```

When receiving URL requests for the server system, like those for local files on the system, the **UseCanonicalName** directive will use the **ServerName** and **Port** directives to generate the host URL server name. When off, it will just use the name supplied by the client request. This can be confusing if the Web server is referenced by one name but uses another, like **www.mytrek.com** used to reference turtle.mytrek.com. **UseCanonicalName** set to on will overcomes this problem, generating the correct local URL.

On Fedora systems, entries have already been made for the standard Web server installation using **/var/www** as your Web site directory. You can tailor your Web site to your own needs by changing the appropriate directives. The **DocumentRoot** directive determines the home directory for your Web pages.

```
DocumentRoot /var/www/html
```

Note: You can also configure Apache to operate as just a proxy and/or cache server. Default proxy and cache server directives are already included in the **httpd.conf** file. The **ProxyRequests** directive turns proxy activity on. Caching can be configured with directives like **CacheRoot** to specify the cache directory, **CachSize** for the cache size (500KB default), and **CacheMaxExpire** to set a time limit on unmodified documents.

Directory-level Configuration

One of the most flexible aspects of Apache is its ability to configure individual directories. With the **Directory** directive, you can define a block of directives that apply only to a particular directory. Such a directive can be placed in the **httpd.conf** or **access.conf** configuration file. You can also use an **.htaccess** file within a particular directory to hold configuration directives. Those directives are then applied only to that directory. The name ".htaccess" is set with the **AccessFileName** directive. You can change this if you want.

```
AccessFileName .htaccess
```

A Directory block begins with a **<Directory** *pathname***>** directive, where *pathname* is the directory to be configured. The ending directive uses the same **<>** symbols, but with a slash preceding the word "Directory": **</Directory>**. Directives placed within this block apply only to the specified directory. The following example denies access to only the **mypics** directory by requests from **www.myvids.com**.

```
<Directory /var/www/mypics>
 Order Deny,Allow
 Deny from www.myvids.com
</Directory>
```

With the **Options** directive, you can enable certain features in a directory, such as the use of symbolic links, automatic indexing, execution of CGI scripts, and content negotiation. The default is the **All** option, which turns on all features except content negotiation (**Multiviews**). The following example enables automatic indexing (**Indexes**), symbolic links (**FollowSymLinks**), and content negotiation (**Multiviews**).

```
Options Indexes FollowSymLinks Multiviews
```

Configurations made by directives in main configuration files or in upper-level directories are inherited by lower-level directories. Directives for a particular directory held in **.htaccess** files and Directory blocks can be allowed to override those configurations. This capability can be controlled by the **AllowOverride** directive. With the **all** argument, **.htaccess** files can override any previous configurations. The **none** argument disallows overrides, effectively disabling the **.htaccess** file. You can further control the override of specific groups of directives. **AuthConfig** enables use of authorization directives, **FileInfo** is for type directives, **Indexes** is for indexing directives, **Limit** is for access control directives, and **Options** is for the options directive.

```
AllowOverride all
```

When given a URL for a directory instead of an HTML file, and when no default web page is in the directory, Apache creates a page on the fly and displays it. This is usually only a listing of the different files in the directory. In effect, Apache indexes the items in the directory for

you. You can set several options for generating and displaying such an index. If **FancyIndexing** is turned on, web page items are displayed with icons and column headers that can be used to sort the listing.

```
FancyIndexing on
```

Access Controls

With access control directives, such as **allow** and **deny**, you can control access to your website by remote users and hosts. The **allow** directive followed by a list of hostnames restricts access to only those hosts. The **deny** directive with a list of hostnames denies access by those systems. The argument **all** applies the directive to all hosts. The **order** directive specifies in what order the access control directives are to be applied. Other access control directives, such as **require**, can establish authentication controls, requiring users to log in. The access control directives can be used globally to control access to the entire site or placed within **Directory** directives to control access to individual directives. In the following example, all users are allowed access:

```
order allow,deny
allow from all
```

Authentication

Your web server can also control access on a per-user or per-group basis to particular directories on your website. You can require various levels for authentication. Access can be limited to particular users and require passwords, or expanded to allow members of a group access. You can dispense with passwords altogether or set up an anonymous type of access, as used with FTP.

To apply authentication directives to a certain directory, you place those directives within either a **Directory** block or the directory's **.htaccess** file. You use the **require** directive to determine what users can access the directory. You can list particular users or groups. The **AuthName** directive provides the authentication realm to the user, the name used to identify the particular set of resources accessed by this authentication process. The **AuthType** directive specifies the type of authentication, such as basic or digest. A **require** directive requires also **AuthType**, **AuthName**, and directives specifying the locations of group and user authentication files. In the following example, only the users **george**, **robert**, and **mark** are allowed access to the **newpics** directory:

```
<Directory /var/www/newpics
     AuthType Basic
     AuthName Newpics
     AuthUserFile /web/users
     AuthGroupFile /web/groups
     <Limit GET POST>
           require users george robert mark
     </Limit>
</Directory>
```

To set up anonymous access for a directory, place the **Anonymous** directive with the user anonymous as its argument in the directory's Directory block or **.htaccess** file. You can also use the **Anonymous** directive to provide access to particular users without requiring passwords from them.

Apache maintains its own user and group authentication files specifying what users and groups are allowed access to which directories. These files are normally simple flat files, such as your system's password and group files. They can become large, however, possibly slowing down authentication lookups. As an alternative, many sites have used database management files in place of these flat files. Database methods are then used to access the files, providing a faster response time. Apache has directives for specifying the authentication files, depending on the type of file you are using. The **AuthUserfile** and **AuthGroupFile** directives are used to specify the location of authentication files that have a standard flat file format. The **AuthDBUserFile** and **AuthDBGroupFile** directives are used for DB database files, and the **AuthDBMGUserFIle** and **AuthDBMGGroupFile** are used for DBMG database files.

The programs htdigest, htpasswd, and dbmmanage are tools provided with the Apache software package for creating and maintaining *user authentication files,* which are user password files listing users who have access to specific directories or resources on your website. The htdigest and htpasswd programs manage a simple flat file of user authentication records, whereas dbmmanage uses a more complex database management format. If your user list is extensive, you may want to use a database file for fast lookups. htdigest takes as its arguments the authentication file, the realm, and the username, creating or updating the user entry. htpasswd can also employ encryption on the password. dbmmanage has an extensive set of options to add, delete, and update user entries. A variety of different database formats are used to set up such files. Three common ones are Berkeley DB2, NDBM, and GNU GBDM. dbmmanage looks for the system libraries for these formats in that order. Be careful to be consistent in using the same format for your authentication files.

User Web sub-sites in user home directories: UserDir

Apache supports user-based sites on server using html files located in user home directories. These are users set up on the Apache server host machine. You use the ~ with the user name to select that user's sub-site. The following will access the user robert's Web sub-site on the **www.turtle.com** site.

```
www.turtle.com/~robert
```

In the user's home directory a public_html directory is set up which holds the html files for that user's site. The **public_html** directory has to have executable permission enabled to allow access by all users, **a+x**. Access will also have to be allowed by all users to the user home directory.

```
chmod a+x public_html
chmod a+x /home/robert
```

The user robert's site would be located at **/home/robert/public_html**.

For Apache to support sites in user home directories, you need to enable the UserDir directive in the **/etc/httpd/conf/httpd.conf** file. UserDir is disabled by default. A commented out enabled version is provided below its entry. You can simple remove or comment out the UserDir disabled entry, then remove the comment symbol (#) from the enabled entry. The UserDir directive is enabled by listing the name to be used for the user Web directories as an option. The default directory is **public_html**.

```
UserDir  public_html
```

If SELinux is enabled, you will also have to allow access by the Web server to the user Web files. These are created as local user files with a user security context. You will have to change the security type for the security context of the user Web file to **httpd_sys_content_t**. This is the same context type used for Web html file in **/var/www**. For a persistent change use **semanage** and the **fcontext** directive with the **-a** and **-t** options. Then run **restorecon**. Add **(/.*)** regular expression to the directory name, and quote it, if you already have some Web files in that directory. If the directory is empty, you don't need it.

```
semanage  fcontext -a -t httpd_sys_content_t "/home/robert/public_html(/.*)"
restorecon  -R /home/robert/public_html
```

Should you want just a temporary change, you can use **chcon**. This change will be lost though if your system relabels or restores contexts.

```
chcon -Rt httpd_sys_content_t  public_html
```

The **httpd.conf** file also includes commented code you can use to secure user Web directories by making them read only. A Directory directive on the public_html directory uses Limit directives with order and allow options to control access.

Log Files

Apache maintains logs of all requests by users to your Web site. By default, these logs include records using the Common Log Format (CLF). The record for each request takes up a line composed of several fields: host, identity check, authenticated user (for logins), the date, the request line submitted by the client, the status sent to the client, and the size of the object sent in bytes.

Webalizer

Reports on Web logs can be generated using the Webalizer tool. Webalizer will display information on your Web site usage. When you run the webalizer command, usage reports will be placed in the **/var/www/html/usage** directory. Access the index page to display a page with links to monthly reports, file:/var/www/html/usage/index.html. Report configuration is specified in the **/etc/webalizer.conf** file. Previous summaries are kept in the /etc/webalizer.history file. Webaliser Apache configuration is located in the **webalizer.conf** file in the **conf.d** directory, **/etc/httpd/conf.d/webalizer.conf**.

Customizing Logs

Using the LogFormat and CustomLog directives, you can customize your log record to add more fields with varying levels of detail. These directives use a format string consisting of field specifiers to determine the fields to record in a log record. You add whatever fields you want and in any order. A field specifier consists of a percent (%) symbol followed by an identifying character. For example, %h is the field specifier for a remote host, %b for the size in bytes, and %s for the status. See the documentation for the mod_log_config module for a complete listing. You should quote fields whose contents may take up more than one word. The quotes themselves must be quoted with a backslash to be included in the format string. The following example is the Common Log Format implemented as a FormatLog directive:

```
FormatLog "%h %l %u %t \"%r\" %s %b"
```

Certain field specifiers in the log format can be qualified to record specific information . The %i specifier records header lines in requests the server receives. The reference for the specific header line to record is placed within braces between the % and the field specifier. For example, User-agent is the header line that indicates the browser software used in the request. To record User-agent header information, use the conversion specifier %{User-agent}i.

To maintain compatibility with NCSA servers, Apache originally implemented AgentLog and RefererLog directives to record User-agent and Referer headers. These have since been replaced by qualified %i field specifiers used for the LogFormat and CustomLog directives. A Referer header records link information from clients, detecting who may have links to your site. The following is an NCSA-compliant log format:

```
"%h %l %u %t \"%r\" %s %b\"%{Referer}i\" \"%{User-agent}i\"".
```

Generating and Managing Log Files

Instead of maintaining one large log file, you can create several log files using the CustomLog or TransferLog directive. This is helpful for virtual hosts where you may want to maintain a separate log file for each host. You use the FormatLog directive to define a default format for log records. The TransferLog then uses this default as its format when creating a new log file. CustomLog combines both operations, enabling you to create a new file and to define a format for it.

```
FormatLog "%h %l %u %t \"%r\" %s %b"
# Create a new log file called myprojlog using the FormatLog format
TransferLog myprojlog
# Create a new log file called mypicslog using its own format
CustomLog mypicslog "%h %l %u %t \"%r\" %s %b"
```

Apache provides two utilities for processing and managing log files: **logresolve** resolves IP addresses in your log file to hostnames; **rotatelogs** rotates log files without having to kill the server. You can specify the rotation time.

Note: The Apache Web server can also provide detailed reports on server activity and configuration, letting you display this information to remote servers. The Location directive server-info will display the configuration details of your Web server, and the server-status directive will show Web processes. The pages server-info and server-status will display the reports, as in http://localhost/server-info. Use the ExtendedStatus directive to enable detailed reports.

Virtual Hosting on Apache

Virtual hosting allows the Apache Web server to host multiple Web sites as part of its own. In effect, the server can act as several servers, each hosted Web site appearing separate to outside users. Apache supports both IP address–based and name-based virtual hosting. IP address–based virtual hosts use valid registered IP addresses, whereas name-based virtual hosts use fully qualified domain addresses. These domain addresses are provided by the host header from the requesting browser. The server can then determine the correct virtual host to use on the basis of the domain name alone. Note that SSL servers require IP virtual hosting. See **http://httpd.apache.org** for more information.

IP based virtual hosting

In the IP address–based virtual hosting method, your server must have a different IP address for each virtual host. The IP address you use is already set up to reference your system. Network system administration operations can set up your machine to support several IP addresses. Your machine could have separate physical network connections for each one, or a particular connection could be configured to listen for several IP addresses at once. In effect, any of the IP addresses can access your system.

You can configure Apache to run a separate daemon for each virtual host, separately listening for each IP address, or you can have a single daemon running that listens for requests for all the virtual hosts. To set up a single daemon to manage all virtual hosts, use **VirtualHost** directives. To set up a separate daemon for each host, also use the **Listen** directive.

Name-Based Virtual Hosting

With name-based virtual hosting, you can support any number of virtual hosts using no additional IP addresses. With only a single IP address for your machine, you can still support an unlimited number of virtual hosts. Such a capability is made possible by the HTTP/1.1 protocol, which lets a server identify the name by which it is being accessed. This method requires the client, the remote user, to use a browser that supports the HTTP/1.1 protocol, as current browsers do (though older ones may not). A browser using such a protocol can send a host header specifying the particular host to use on a machine.

```
ServerName turtle.mytrek.com
NameVirtualHost 192.168.1.5

<VirtualHost 192.168.1.5>
 ServerName www.mypics.com
 ServerAdmin webmaster@mail.mypics.com
 DocumentRoot /var/www/mypics/html
 ErrorLog /var/www/mypics/logs/error_log
 ...
</VirtualHost>

<VirtualHost 192.168.1.5>
 ServerName www.myproj.org
 ServerAdmin webmaster@mail.myproj.org
 DocumentRoot /var/www/myproj/html
 ErrorLog /var/www/myproj/logs/error_log
 ....
</VirtualHost>
```

To implement name-based virtual hosting, use a **VirtualHost** directive for each host and a **NameVirtualHost** directive to specify the IP address you want to use for the virtual hosts. If your system has only one IP address, you need to use that address. Within the **VirtualHost** directives, you use the **ServerName** directive to specify the domain name you want to use for that host. Using **ServerName** to specify the domain name is important to avoid a DNS lookup. A DNS lookup failure disables the virtual host. The **VirtualHost** directives each take the same IP address specified in the **NameVirtualHost** directive as their argument. You use Apache directives within the **VirtualHost** blocks to configure each host separately. Name-based virtual hosting uses the domain

name address specified in a host header to determine the virtual host to use. If no such information exists, the first host is used as the default. The following example implements two name-based virtual hosts. Here, **www.mypics.com** and **www.myproj.org** are implemented as name-based virtual hosts instead of IP-based hosts:

If your system has only one IP address, implementing virtual hosts prevents access to your main server with that address. You could no longer use your main server as a Web server directly; you could use it only indirectly to manage your virtual host. You could configure a virtual host to manage your main server's Web pages. You would then use your main server to support a set of virtual hosts that would function as Web sites, rather than the main server operating as one site directly. If your machine has two or more IP addresses, you can use one for the main server and the other for your virtual hosts. You can even mix IP-based virtual hosts and name-based virtual hosts on your server. You can also use separate IP addresses to support different sets of virtual hosts. You can further have several domain addresses access the same virtual host. To do so, place a **ServerAlias** directive listing the domain names within the selected **VirtualHost** block.

```
ServerAlias www.mypics.com www.greatpics.com
```

Dynamic Virtual Hosting

If you have implemented many virtual hosts on your server that have the same configuration, you can use a technique called dynamic virtual hosting to have these virtual hosts generated dynamically. The code for implementing your virtual hosts becomes much smaller, and as a result, your server accesses them faster. Adding yet more virtual hosts becomes a simple matter of creating appropriate directories and adding entries for them in the DNS server.

To make dynamic virtual hosting work, the server uses commands in the **mod_vhost_alias** module (supported in Apache version 1.3.6 and up) to rewrite both the server name and the document root to those of the appropriate virtual server (for older Apache versions before 1.3.6, you use the **mod_rewrite** module). Dynamic virtual hosting can be either name-based or IP-based. In either case, you have to set the **UseCanonicalName** directive in such a way as to allow the server to use the virtual hostname instead of the server's own name. For name-based hosting, you simply turn off **UseCanonicalName**. This allows your server to obtain the hostname from the host header of the user request. For IP-based hosting, you set the **UseCanonicalName** directive to DNS. This allows the server to look up the host in the DNS server.

```
UseCanonicalName Off
UseCanonicalName DNS
```

You then have to enable the server to locate the different document root directories and CGI bin directories for your various virtual hosts. You use the **VirtualDocumentRoot** directive to specify the template for virtual hosts' directories. For example, if you place the different host directories in the **/var/www/hosts** directory, you can then set the **VirtualDocumentRoot** directive accordingly.

```
VirtualDocumentRoot /var/www/hosts/%0/html
```

The %0 will be replaced with the virtual host's name when that virtual host is accessed. It is important that you create the dynamic virtual host's directory using that host's name. For example, for a dynamic virtual host called **www.mygolf.org**, you first create a directory named **/var/www/hosts/www.mygolf.org**, then create subdirectories for the document root and CGI

programs, as in **/var/www/hosts/www.mygolf.org/html**. For the CGI directory, use the **VirtualScriptAlias** directive to specify the CGI subdirectory you use.

```
VirtualScriptAlias /var/www/hosts/%0/cgi-bin
```

A simple example of name-based dynamic virtual hosting directives follows:

```
UseCanonicalName Off
VirtualDocumentRoot /var/www/hosts/%0/html
VirtualScriptAlias /var/www/hosts/%0/cgi-bin
```

A request for **www.mygolf.com/html/mypage** evaluates to

```
/var/www/hosts/www.mygolf.com/html/mypage
```

A simple example of dynamic virtual hosting is shown here:

```
UseCanonicalName Off

NameVirtualHost 192.168.1.5

<VirtualHost 192.168.1.5>
 ServerName www.mygolf.com
 ServerAdmin webmaster@mail.mygolf.com
 VirtualDocumentRoot /var/www/hosts/%0/html
 VirtualScriptAlias /var/www/hosts/%0/cgi-bin
 ...
</VirtualHost>
```

To implement IP-based dynamic virtual hosting instead, set the **UseCanonicalName** to DNS instead of Off.

```
UseCanonicalName DNS
VirtualDocumentRoot /var/www/hosts/%0/html
VirtualScriptAlias /var/www/hosts/%0/cgi-bin
```

Interpolated Strings

The **mod_vhost_alias** module supports various interpolated strings, each beginning with a **%** symbol and followed by a number. As you have seen, **%0** references the entire web address. **%1** references only the first segment, **%2** references the second, **%-1** references the last part, and **%2+** references from the second part on. For example, to use only the second part of a web address for the directory name, use the following directives:

```
VirtualDocumentRoot /var/www/hosts/%2/html
VirtualScriptAlias /var/www/hosts/%2/cgi-bin
```

In this case, a request made for **www.mygolf.com/html/mypage** uses only the second part of the web address. This would be "mygolf" in **www.mygolf.com**, and would evaluate to

```
/var/www/hosts/mygolf/html/mypage
```

If you used **%2+** instead, as in **/var/www/hosts/%2/html**, the request for **www.mygolf.com/html/mypage** would evaluate to

```
/var/www/hosts/mygolf.com/html/mypage
```

The same method works for IP addresses, where `%1` references the first IP address segment, `%2` references the second, and so on.

Logs for Virtual Hosts

One drawback of dynamic virtual hosting is that you can set up only one log for all your hosts. However, you can create your own shell program to simply cut out the entries for the different hosts in that log.

```
LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon
```

Note: Apache also supports IP address–based virtual hosting. Your server must have a different IP address for each virtual host. Your machine can have separate physical network connections for each one. You can configure Apache to run a separate daemon for each virtual host, separately listening for each IP address, or you can have a single daemon running that listens for requests for all the virtual hosts. To set up a single daemon to manage all virtual hosts, use **VirtualHost** directives. To set up a separate daemon for each host, also use the **Listen** directive.

Server Side Includes

Server-side includes (SSIs) are designed to provide a much more refined control of your Web site content, namely the Web pages themselves. Server-side includes are Apache directives placed within particular Web pages as part of the page's HTML code. You can configure your Apache Web server to look for SSI directives in particular Web pages and execute them. First, you have to use the Options directive with the include option to allow SSI directives,

```
Options Includes
```

You need to instruct the server to parse particular Web pages. The easiest way to enable parsing is to instruct Apache to parse HTML files with specified extensions. Usually, the extension .shtml is used for Web pages that have SSI directories. In fact, in the default Apache configuration files, you can find the following entry to enable parsing for SSI directives in HTML files. The AddType directive here adds the .shtml type as an HTML type of file, and the AddHandler directive specifies that .shtml files are to be parsed (server-parsed):

```
# To use server-parsed HTML files
AddType text/html .shtml
AddHandler server-parsed .shtml
```

Instead of creating a separate type of file, you can use the XBitHack directive to have Apache parse any executable file for SSI directives. In other words, any file with execute permission will be parsed for SSI directives.

SSI directives operate much like statements in a programming language. You can define variables, create loops, and use tests to select alternate directives. An SSI directive consists of an element followed by attributes that can be assigned values. The syntax for a SSI directive is shown here:

```
<!--#element attribute=value ... -->
```

You can think of an element as operating much like a command in a programming language and attributes as its arguments. For example, to assign a value to a variable, you use the set element with the variable assignment as its attribute. The **if** directive displays any following text on the given Web page. The **if** directive takes as its attribute **expr**, which is assigned the expression to test. The test is able to compare two strings using standard comparison operators such as **<=**, **!=**, or **=**. Variables used in the test are evaluated with the **$** operator.

```
<!--#set myvar="Goodbye" -->
<!--#if expr="$myvar = Hello" -->
```

Other helpful SSI elements are exec, which executes CGI programs, or shell commands, which read the contents of a file into the Web page and also execute CGI files. The echo element displays values such as the date, the document's name, and the page's URL. With the config element, you can configure certain values, such as the date or file size.

PHP

PHP (PHP: Hypertext Preprocessor) is a scripting language designed for use in Web pages. PHP-enabled pages allow you to create dynamic Web pages that can perform tasks instead of just displaying data. PHP is an official project of the Apache Software Foundation. You can find out more about PHP at **www.php.net**.

Unlike CGI programs, which are executed separately from a Web page, PHP commands are embedded as tags within the page itself, much as SSI commands are. PHP support to interpret and execute these commands is provided directly by the Web server. This embedded support is enabled in Apache with the **mod_php** module (**/etc/httpd/conf.d/php.conf** configuration file). Instead of having to separately construct programs to be invoked and run outside the Web server, with PHP, such commands are embedded within a Web page and run by the Web server. The Web server maintains complete control at all times whenever tasks are being performed. It is possible, however, to implement PHP in a CGI mode, where PHP pages are constructed as separate programs, invoked by a Web page much as a Perl-based CGP program is.

PHP has flexible and powerful programming capabilities on the same level as C and Perl. As in those languages, you can create control structures such as if statements and loops. In addition, PHP has capabilities specifically suited to Web page tasks. PHP can interact directly with databases such as Oracle, MySQL, and IBM DB2. It can easily interact with all the standard protocols, such as IMAP, LDAP, HTTP, and POP3. It even has text processing abilities such as interpreting regular expressions and displaying XML documents. There are also extensions for searches, compression tools like gzip, and language translations. PHP supports a massive collection of possible operations. Check its Web site for a complete listing, as well as online manuals and tutorials.

Web Server Security: SSL

Web server security deals with two different tasks: protecting your Web server from unauthorized access, and providing security for transactions carried out between a Web browser client and your Web server. To protect your server from unauthorized access, you use a proxy server such as Squid. Squid is a GNU proxy server often used with Apache on Linux systems. (See Chapter 29 for an explanation of the Squid server.) Apache itself has several modules that provide security capabilities. These include **mod_access** for mandatory controls; **mod_auth**, **mod_auth_db**, **mod_auth_digest**, and **mod_auth_dbm**, which provide authentication support;

and **mod_auth_anon** for anonymous FTP-like logging (see previous sections on access control and authentication).

To secure transmissions, you need to perform three tasks. You have to verify identities, check the integrity of the data, and ensure the privacy of the transmission. To verify the identities of the hosts participating in the transmission, you perform authentication procedures. To check the integrity of the data, you add digital signatures containing a digest value for the data. The digest value is a value that uniquely represents the data. Finally, to secure the privacy of the transmission, you encrypt it. Transactions between a browser and your server can then be encrypted, with the browser and your server alone able to decrypt the transmissions. The protocol most often used to implement secure transmissions with Linux Apache Web servers is the Secure Sockets Layer (SSL) protocol, which was originally developed by Netscape for secure transactions on the Web.

Like the Secure Shell (SSH) and the GNU Privacy Guard, SSL uses a form of public- and private-key encryption for authentication. Data is encrypted with the public key but can be decrypted only with the private key. Once the data is authenticated, an agreed-upon cipher is used to encrypt it. Digital signatures encrypt an MD5 digest value for data to ensure integrity. Authentication is carried out with the use of certificates of authority. Certificates identify the different parties in a secure transmission, verifying that they are who they say they are. A Web server will have a certificate verifying its identity, verifying that it is the server it claims to be. The browser contacting the server will also have a certificate identifying who it is. These certificates are, in turn, both signed by a certificate authority, verifying that they are valid certificates. A certificate authority is an independent entity that both parties trust.

A certificate contains the public key of the particular server or browser it is given to, along with the digital signature of the certificate authority and identity information such as the name of the user or company running the server or browser. The effectiveness of a certificate depends directly on the reliability of the certificate authority issuing it. To run a secure Web server on the Internet, you should obtain a certificate from a noted certificate authority such as VeriSign. A commercial vendor such as Stronghold can do this for you. Many established companies already maintain their own certificate authority, securing transmissions within their company networks. An SSL session is set up using a handshake sequence in which the server and browser are authenticated by exchanging certificates, a cipher is agreed upon to encrypt the transmissions, and the kind of digest integrity check is chosen. There is also a choice in the kind of public key encryption used for authentication, either RSA or DSA. For each session, a unique session key is set up that the browser and server use.

A free open source version of SSL called OpenSSL is available for use with Apache (see **www.openssl.org**). It is based on SSLeay from Eric A. Young and Tim J. Hudson. However, U.S. government restrictions prevent the Apache Web server from being freely distributed with SSL capabilities built in. You have to separately obtain SSL and update your Apache server to incorporate this capability.

The U.S. government maintains export restrictions on encryption technology over 40 bits. SSL, however, supports a number of ciphers using 168-, 128-, and 40-bit keys (128 is considered secure, and so by comparison the exportable 40-bit versions are useless). This means that if Apache included SSL, it could not be distributed outside the United States. Outside the United States, however, there are projects that do distribute SSL for Apache using OpenSSL. These are free for noncommercial use in the United States, though export restrictions apply. The Apache-SSL project freely distributes Apache with SSL built in, apache+ssl. You can download this from their Web site

at www.apache-ssl.org (though there are restrictions on exporting encryption technology, there are none on importing it). In addition, the **mod_ssl** project provides an SSL module with patches you can use to update your Apache Web server to incorporate SSL (www.modsll.org). mod_ssl is free for both commercial and noncommercial use under an Apache-style license. Red Hat and Fedora include the mod_ssl module with its distribution in the mod_ssl package (**/etc/httpd/conf.d/ssl.conf** configuration file).

The mod_ssl implementation of SSL provides an alternate access to your Web server using a different port (443) and a different protocol, https. In effect, you have both an SSL server and a nonsecure version. To access the secure SSL version, you use the protocol https instead of http for the Web server's URL address. For example, to access the SSL version for the Web server running at **www.mytrek.com**, you would use the protocol https in its URL, as shown here:

```
https://www.mytrek.com
```

You can configure mod_ssl using a number of configuration directives in the Apache configuration file, **smb.conf**. On Fedora, the default configuration file installed with Apache contains a section for the SSL directives along with detailed comments. Check the online documentation for mod_ssl at www.modssl.org for a detailed reference listing all the directives. There are global, server-based, and directory-based directives available.

In the Fedora **smb.conf** file, the inclusion of SSL directives are controlled by IfDefine blocks enabled by the HAVE_SSL flag. For example, the following code will load the SSL module:

```
<IfDefine HAVE_SSL>
LoadModule ssl_module     modules/libssl.so
</IfDefine>
```

The SSL version for your Apache Web server is set up in the **smb.conf** file as a virtual host. The SSL directives are enabled by an ifDefine block using the HAVE_SSL flag. Several default directives are implemented such as the location of SSL key directories and the port that the SSL version of the server will listen on (443). Others are commented out. You can enable them by removing the preceding # symbol, setting your own options. Several of the directives are shown here:

```
<IfDefine HAVE_SSL>
## SSL Virtual Host Context

#  Server Certificate:
SSLCertificateFile /etc/httpd/conf/ssl.crt/server.crt

#  Server Private Key:
SSLCertificateKeyFile /etc/httpd/conf/ssl.key/server.key

#  Certificate Authority (CA):
#SSLCACertificatePath /etc/httpd/conf/ssl.crt
#SSLCACertificateFile /etc/httpd/conf/ssl.crt/ca-bundle.crt
```

In the **/etc/httpd/conf** directory, mod_ssl will set up several SSL directories that will contain SSL authentication and encryption keys and data. The **ssl.crt** directory will hold certificates for the server. The **ssl.key** directory holds the public and private keys used in authentication encryption. Revocation lists for revoking expired certificates are kept in **ssl.crl**. The **ssl.csr**

directory holds the certificate signing request used to request an official certificate from a certificate authority. **ssl.prm** holds parameter files used by the DSA key encryption method. Check the README files in each directory for details on the SSL files they contain.

The mod_ssl installation will provide you with a demonstration certificate called snakeoil that you can use to test your SSL configuration. When you have an official certificate, you can install it with the **`make certificate`** command within the **ssl.crt** directory. This will overwrite the **server.crt** server certificate file.

fedora

23. News and Database Services

Linux Overview
Fedora Linux
Fedora Documentation
Fedora 10 changes
Standard Fedora features
Fedora Live DVD/CD
Fedora 10 Desktop Look and Feel:
Open Source Software
Operating Systems and Linux
A Short History of Linux and Unix
Red Hat Enterprise Linux

Newsgroup severs are more rare, used for setting up newsgroups for local networks or for supporting the Internet's Usenet News service. Database servers are becoming more common for managing large collections of data on local networks as well as for Internet services.

News Servers

News servers provide Internet users with Usenet news services. They have their own TCP/IP protocol, the Network News Transfer Protocol (NNTP). On most Linux systems, the InterNetNews (INN) news server provides news services (**www.isc.org**). In addition, servers exist that provide better access to Internet resources.

The InterNetNews (INN) news server accesses Usenet newsfeeds, providing news clients on your network with the full range of newsgroups and their articles. Newsgroup articles are transferred using NNTP, and servers that support this protocol are known as *NNTP servers*. INN was written by Rich Salz and is currently maintained and supported by the Internet Software Consortium (ISC). You can download current versions from its website at **www.isc.org**. INN is also included with most Linux distributions. The documentation directory for INN in **/usr/share/doc** contains extensive samples. The primary program for INN is the **innd** daemon.

There are two versions of INN, a smaller INN used for local networks, and a much more complex INN2 used for large networks. INN also includes several support programs to provide maintenance and crash recovery and to perform statistical analysis on server performance and usage. cleanfeed implements spam protection, and innreport generates INN reports based on logs. INN also features a very strong filter system for screening unwanted articles.

Note: Leafnode is an NNTP news server designed for small networks that may have slow connections to the Internet. You can obtain the Leafnode software package along with documentation from its website at **www.leafnode.org**.

Database Servers: MySQL and PostgreSQL

Two fully functional database servers are included with most Linux distributions, MySQL and PostgreSQL. MySQL is by far the more popular of the two, though PostgreSQL is noted for providing more features. This chapter will cover how to set up and manage a MySQL database and will offer a brief introduction to PostgreSQL. You can learn more about these products through the sites listed in Table 23-1.

Database	Resource
MySQL	www.mysql.com
PostgreSQL	www.postgresql.org

Table 23-1: Database Resources

Relational Database Structure

MySQL and PostgreSQL both use a relational database structure. Essentially, this means data is placed in tables, with identifier fields used to relate the data to entries in other tables. Each row in the table is a record, each with a unique identifier, like a record number. The connections between records in different tables are implemented by special tables that associate the unique

identifiers from records in one table with those of another. Relational database theory and implementation are subjects beyond the scope of this chapter.

A simple, single-table database has no need for a unique identifier. A simple address book listing names and addresses is an example of a single-table database. However, most databases access complex information of different types, related in various ways. Instead of having large records with repeated information, you divide the data in different tables, each holding the unique instance of the data. This way, data is not repeated; you have only one table that holds a single record for person's name, rather than repeating that person's name each time the data references him or her. The relational organization then takes on the task of relating one piece of data to another. This way, you can store a great deal of information using relatively small database files.

Though there are many ways to implement a relational database, a simple rule of thumb is to organize data into tables where you have a unique instance of each item of data. Each record is given a unique identifier, usually a number. To associate the records in one table with another, you create tables that associate their identifiers.

The SQL query language is the language used by most relational database management systems (RDBMSs), including both MySQL and PostgreSQL. Though many RDBMSs use administrative tools to manage databases, on Linux MySQL and PostgreSQL, you still have to use the SQL commands directly. The following command will create the database:

```
CREATE DATABASE myphotos
```

Before performing any operations on a database, you first access it with the USE command.

```
USE myphotos
```

The tables are created using the CREATE TABLE command; the fields for each table are listed within parentheses following the table name. For each field, you need to specify a name, data type, and other options, such as whether it can have a null value or not.

```
CREATE TABLE names (
    personid INT(5) UNSIGNED NOT NULL,
    name VARCHAR(20) NOT NULL,
    street VARCHAR(30) NOT NULL,
    phone CHAR(8)
     );
```

MySQL

MySQL is structured on a client/server model with a server daemon (**mysqld**) filling requests from client programs. MySQL is designed for speed, reliability, and ease of use. It is meant to be a fast database management system for large databases and, at the same time, a reliable one, suitable for intensive use. To create databases, you use the standard SQL language. User access can be controlled by assigning privileges.

On Fedora you can install MySQL server and client packages, along with numerous MySQL configuration packages for certain services like Postfix, Exim, and Apache . Packages to install are **mysql-client**, **mysql-common**, and **mysql-server**. Documentation is held in the **mysql-doc** package and installed at **/usr/share/doc/mysql-doc**.

MySQL Configuration

The MySQL supports three different configuration files, one for global settings, and another for server-specific settings, and an optional one for user-customized settings.

- The **/etc/my.cnf** configuration file is used for global settings applied to both clients and servers. The **/etc/my.cnf** file provides information such as the data directory (**/var/lib/mysql**) and log files (**/var/log/mysql**) locations, as well as the server base directory (**/var/lib**).
- The **/var/lib/mysql/my.cnf** file is used for server settings only.
- The **.my.cnf** file allows users to customize their access to MySQL. It is located in a user's home directory. Note that this is a dot file.

Sample configuration **my.cnf** files can be found in the **mysql-server** directory in **/usr/share/doc**. The **/usr/share/doc/mysql-server** directory lists configurations for small, medium, large, and huge implementations. The administrative manual is located in the **mysql** directory for **/usr/share/doc**. It is in the info format. Use **`info mysql`** to start it and the arrow and ENTER keys to move through the menus. Here you can find more information about different options.

Global Configuration:/etc/my.cnf

MySQL specifies options according to different groups, usually the names of server tools. The options are arranged in group segments. The group name is placed within brackets, and options applied to it follow. The default **/etc/my.cnf** file is shown here:

```
[mysqld]
user=mysql
datadir=/var/lib/mysql
pid-file = /var/run/mysqld/mysqld.pid
socket = /var/run/mysqld/mysqld.sock
port = 3306
basedir = /usr
datadir = /var/lib/mysql
tmpdir = /tmp
language = /usr/share/mysql/English
```

MySQL global options are listed in the **/etc/my.cnf** file. Options are set up according to groups that control different behaviors of the MySQL server: **`mysqld`** for the daemon and **`safe_mysqld`** for the MySQL startup script. The **`datadir`** directory, **/var/lib/mysql**, is where your database files will be placed. Server tools and daemons are located in the **`basedir`** directory, **/usr**, and the user that MySQL will run as has the name **mysql**, as specified in the **`user`** option.

A client group will set up options to be sent to clients, such as the port and socket to use to access the MySQL database.

```
[client]
port=3306
socket=/var/lib/mysqld/mysqld.sock
```

To see what options are currently set for both client and server, you run **`mysqld`** directly with the **`--help`** option.

```
/usr/libexec/mysqld --help
```

User Configuration: .my.cnf

Users who access the database server will have their own configuration file in their home directory: **.my.cnf**. Here the user can specify connection options such as the password used to access the database and the connection timeouts.

```
[client]
password=mypassword

[mysql]
no-auto-rehash
set-variable = connect_timeout=2

[mysql-hotcopy]
interactive-timeout
```

MySQL Tools

MySQL provides a variety of tools (as shown in Table 23-2), including server, client, and administrative tools. Backups can be handled with the **mysldump** command. The **mysqlshow** command will display a database, just as issuing the SQL command SELECT *.* does, and **mysqlimport** can import text files, just like LOAD INFILE.

Command	Description
mysqld	MySQL server
mysql	MySQL client
mysqladmin	Creates and administers databases
mysqldump	Database backup
mysqlimport	Imports text files
mysqlshow	Displays databases

Table 23-2: MySQL Commands

MySQL Management with mysql and mysqladmin

To manage your MySQL database, you use **mysql** as the **root** user. The **mysql** client starts up the MySQL monitor. As the root user, you can enter administrative commands to create databases and database tables, add or remove entries, as well as carry out standard client tasks such as displaying data.

Log in as the root user and open a terminal window. Then enter the **mysql** command. This will start a MySQL monitor shell with a **mysql>** prompt. Be sure to end your commands with a semicolon; otherwise, the monitor will provide an indented arrow prompt waiting for added arguments. In the monitor, the semicolon, not the ENTER key, ends commands.

```
 # mysql -u root -p
mysql>
```

If you have set up a MySQL root user, you can use the **-u** root with the **-p** option. You will be prompted for a password.

```
# mysql -u root -p
```

Once the **mysql** client has started, you can use the **status** command to check the status of your server and **show databases** to list current databases.

```
mysql> status;
mysql> show databases;
```

Initially two databases set up by MySQL for its own management are displayed: mysql and test. The mysql database holds MySQL user information, and the test database is used to test the server.

PostgreSQL

PostgreSQL is based on the POSTGRESQL database management system, though it uses SQL as its query language. POSTGRESQL is a next-generation research prototype developed at the University of California, Berkeley. You can find more information on it from the PostgreSQL website at **www.postgresql.org**. PostgreSQL is an open source project, developed under the GPL license.

PostgreSQL is often used to provide database support for Internet servers with heavy demands, such as web servers. With a few simple commands, you can create relational database tables. Use the **createuser** command to create a PostgreSQL user that you can then log in to the server with. You can then create a database with the **createdb** command and construct relational tables using the **create table** directive. With an **insert** command, you can add records and then view them with the **select** command. Access to the server by remote users is controlled by entries in the **pg_hba.conf** file located in PostgreSQL directory, usually **/var/lib/pgsql**.

Note: The search and indexing server ht://Dig enables document searches of web and FTP sites (**www.htdig.org**). With it, you can index documents and carry out complex search requests.

Part 5: Shared Resources

fedora

24. Print Services

Print services have become an integrated part of every Linux system. They allow you to use any printer on your system or network.

Once treated as devices attached to a system directly, printers are now treated as network resources managed by print servers. In the case of a single printer attached directly to a system, the networking features become transparent and the printer appears as just one more device. On the other hand, you could easily use a print server's networking capability to let several systems use the same printer. Although printer installation is almost automatic on most Linux distributions, it helps to understand the underlying process. Printing sites and resources are listed in Table 24-1.

The Common Unix Printing System (CUPS) provides printing services. It is freely available under the GNU Public License. Though it is now included with most distributions, you can also download the most recent source-code version of CUPS from **www.cups.org**, which provides detailed documentation on installing and managing printers. CUPS is based on the Internet Printing Protocol (IPP), which was designed to establish a printing standard for the Internet (for more information, see **www.pwg.org/ipp**). Whereas the older line printer (LPD)based printing systems focused primarily on line printers, an IPP-based system provides networking, PostScript, and web support. CUPS works like an Internet server and employs a configuration setup much like that of the Apache web server. Its network support lets clients directly access printers on remote servers, without having to configure the printers themselves. Configuration needs to be maintained only on the print servers.

If you cannot find the drivers for your printer, you may be able to download them from OpenPrinting database at **http://linux-foundation.org/en/OpenPrinting**. The site maintains an extensive listing of drivers. The Fedora **system-config-printer** tool can now download the most recent drivers from the OpenPrinting site.

Resource	Description
www.cups.org	Common Unix Printing System
www.pwg.org/ipp	Internet Printing Protocol
http://linux-foundation.org/en/OpenPrinting	OpenPrinting, print drivers

Table 24-1: Print Resources

Printer Services: CUPS

CUPS is the primary print server for most Linux distributions. GNOME provides integrated support for CUPS, allowing GNOME-based applications to directly access CUPS printers.

Once you have installed your printers and configured your print server, you can print and manage your print queue using print clients. There are a variety of printer clients available for the CUPS server, GNOME print manager, the CUPS configuration tool, and various line printing tools like **lpq** and **lpc**. These are described in further detail later in this chapter. The CUPS configuration tool is a web-based configuration tool that can also manage printers and print jobs (open your browser and enter the URL **http://localhost:631**). A web page is displayed with entries for managing jobs, managing printers, and administrative tasks. Select the Manage Jobs entry to remove or reorder jobs you have submitted.

Note: Line Printer, Next Generation (LPRng) was the traditional print server for Linux and Unix systems, but it has since been dropped from Fedora. You can find out more about LPRng at http://sourceforge.net/projects/lprng.

Printer Devices and Configuration

Before you can use any printer, you first have to install it on a Linux system on your network. A local printer is installed directly on your own system. This involves creating an entry for the printer in a printer configuration file that defines the kind of printer it is, along with other features such as the device file and spool directory it uses. On CUPS, the printer configuration file is **/etc/cups/printers.conf**. Installing a printer is fairly simple: determine which device file to use for the printer and the configuration entries for it.

Printer Device Files

Linux dynamically creates the device names for printers that are installed. For parallel printers, the device names will be **lp0**, **lp1**, and **lp2**, depending on how many parallel printers are connected. The number used in these names corresponds to a parallel port on your PC; **lp0** references the LPT1 parallel port and **lp1** references the LPT2 parallel port. Serial printers will use serial ports, referenced by the device files like **ttyS0**, **ttyS1**, **ttyS2**, and so on. USB-connected printers will have a Hardware Abstract Layer (HAL) device connection. HAL is designed for removable devices that can easily be attached to other connections and still be recognized.

Spool Directories

When your system prints a file, it makes use of special directories called *spool directories*. A *print job* is a file to be printed. When you send a file to a printer, a copy of it is made and placed in a spool directory set up for that printer. The location of the spool directory is obtained from the printer's entry in its configuration file. On Linux, the spool directory is located at **/var/spool/cups** under a directory with the name of the printer. For example, the spool directory for the **myepson** printer would be located at **/var/spool/cups/myepson**. The spool directory contains several files for managing print jobs. Some files use the name of the printer as their extension. For example, the **myepson** printer has the files **control.myepson**, which provides printer queue control, and **active.myepson** for the active print job, as well as **log.myepson,** which is the log file.

Server script

A **cups** startup script is installed in the **/etc/rc.d/init.d** directory. You can start, stop, and restart CUPS using the **service** command and the **cups** script or **system-config-services** (System | Administration | Services). When you make changes or install printers, be sure to restart CUPS to have your changes take effect. You can use the following command:

```
service cups restart
```

Printer Install and Configuration

There are several tools available for installing CUPS printers. The easiest method is to use Fedora system-config-printer tool. Alternatively you can use the CUPS Web browser-based configuration tools, included with the CUPS software. Finally you can just edit the CUPS printer configuration files directly.

Whenever you first attach a local printer, like a USB printer, you will be asked to perform basic configuration such as confirming the make and model. Removable local printers are managed by udev and HAL, tools designed to automatically detect and configure removable devices. A message will appear like that shown in Figure 24-1, as soon as you connect your USB printer.

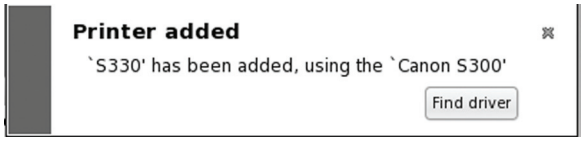


Figure 24-1: Printer detection notification

If a driver is available for your printer, it will be selected automatically for you. Should you want to use a different driver, or printer is being detected incorrectly, you can click on the Find diver button to select the driver. A Choose Driver window opens up where you can enter in the printer configuration information (see Figure 24-2).

Figure 24-2: Change Printer driver

system-config-printer

To later change your configuration or to add a remote printer, you can use the printer configuration tool, system-config-printer. This utility enables you to select the appropriate driver for your printer, as well as set print options such as paper size and print resolutions. You can configure a printer connected directly to your local computer or a printer on a remote system on your network. You can start system-config-printer by selecting the Printing entry in the System | Administration menu.

Fedora 10 uses a more recent version of system-config-printer that has a slightly different interface. All the printer information and configuration formats are the same. You still have the same set of dialogs and panes for setting up a printer and displaying printer configuration. Instead of two panes you have a single window with icons for installed printers (see Figure 24-3).

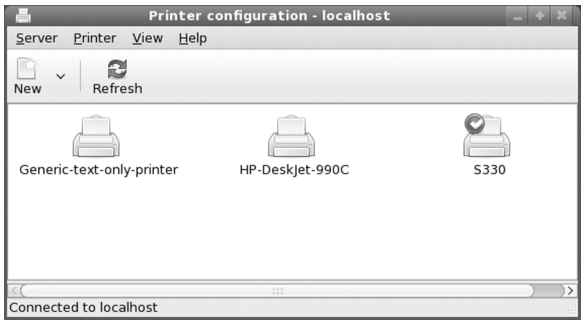


Figure 24-3: system-config-printer tool

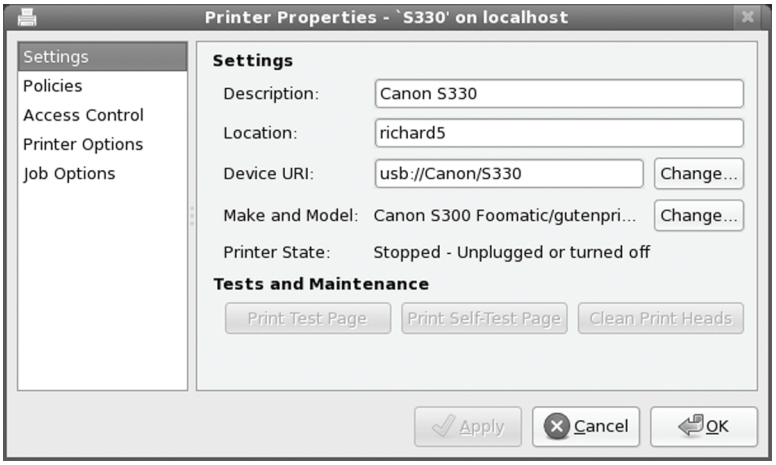


Figure 24-4: Printer properties window

To see the printer settings such as printer and job options, access controls, and policies, double-click on the printer icon or right-click and select Properties. The Printer Properties window opens up with five panes: Settings, Policies, Access Control, Printer Options, and Job Options (see Figure 24-4).

The Printer configuration window Printer menu lets you rename the printer, enable or disable it, and make it a shared printer. Select the printer icon and then click the Printer menu (see Figure 24-5). The Delete entry will remove a printer configuration. Use the Set As Default entry to make the printer a system-wide or personal default printer.

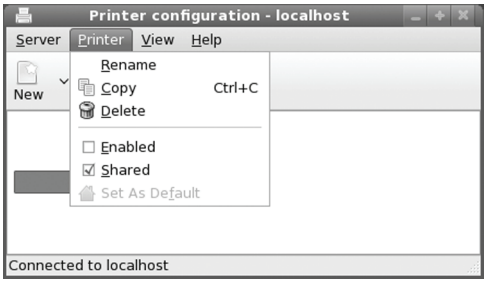


Figure 24-5: Printer configuration window Printer menu

Note: You can perform all administrative tasks from the command line using the `lpadmin` command. See the CUPS documentation for more details.

The Printer icon menu is accesses by right-clicking on the printer icon (see Figure 24-6). It adds entries for accessing the printer properties and viewing the print queue. If the printer is already a default, there is no Set As Default entry. The properties entry opens the printer properties window for that printer.

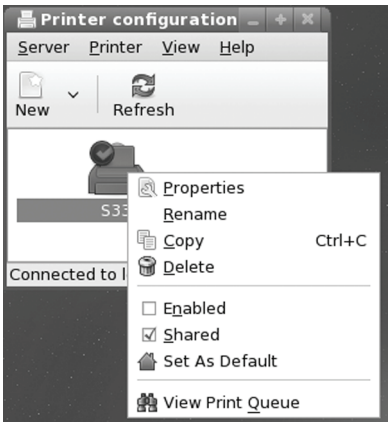


Figure 24-6: Printer icon menu

The View Print Queue entry opens the Document print status window listing jobs for that printer. You can change the queue position as well as stop or delete jobs (see Figure 24-7). From the View menu you can choose to display printed jobs and the printer status. You will be notified if a job should fail. Click the Diagnose button to start the printing trouble- shooter to find the reason.

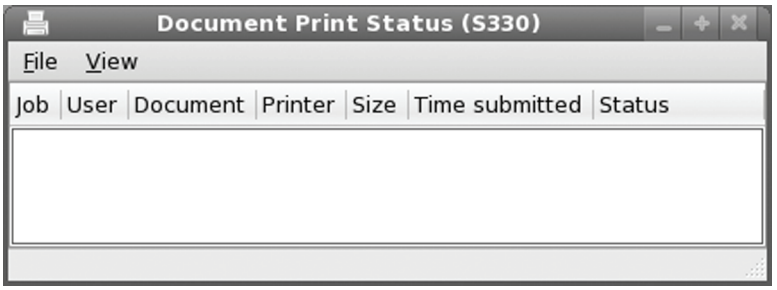


Figure 24-7: Printer queue

To check the server settings, select Settings from the Server menu. This opens a new window showing the CUPS printer server settings (see Figure 24-8). The Advanced button opens a dialog where you can set job history and browse server options. For job history you can choose to keep just a record of print jobs or the actually files, thereby allowing reprinting. Browse server lets you select certain servers on your network for which you want print queues displayed.

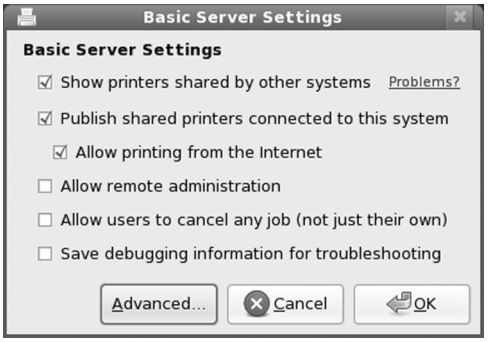


Figure 24-8: Server Settings

To select a particular CUPS server, select the Connect entry in the Server menu. This opens a "Connect to CUPS Server" window with a drop down menu listing all current CUPS servers from which to choose (see Figure 24-9).

Figure 24-9: Selecting a CUPS server

Editing Printer Configuration

To edit an installed printer, double click its icon in the Printer configuration window, or right-click and select the Properties entry. This opens a Printer Properties window for that printer. A sidebar lists configuration panes. Click on one to display that pane. There are configuration entries for Settings, Policies, Access Control, Printer Options, and Job Control (see Figure 24-10).

Once you have made your changes, you can click Apply to save your changes and restart the printer daemon. You can test your printer with a PostScript, A4, or ASCII test sheet selected from the Test menu.

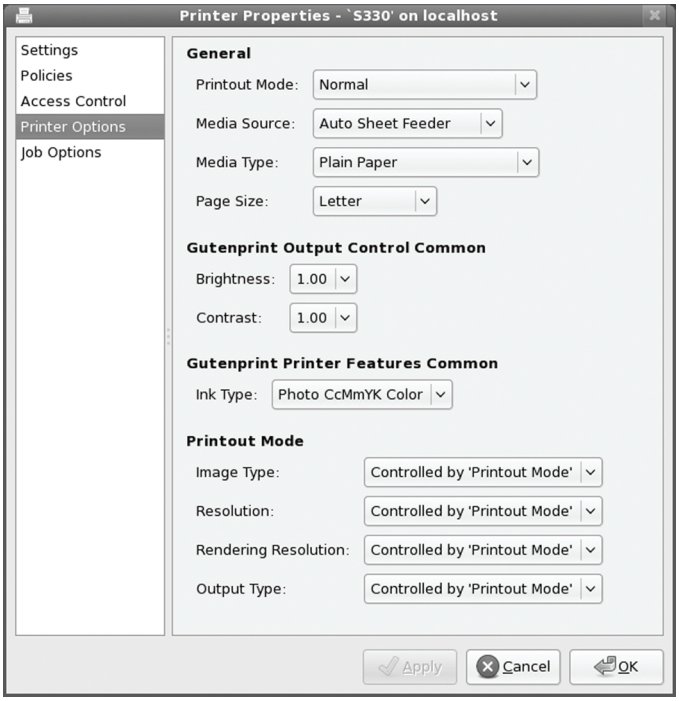


Figure 24-10: Printer Options pane

On the Settings pane you can change configuration settings like the driver and the printer name, enable or disable the printer, or specify whether to share it or not (see Figure 24-4). Should you need to change the selected driver, click on the Change button next to the Make and Model entry. This will open printer model and driver windows like those described in the Add new printer manually section. There you can specify the model and driver you want to use, even loading your own driver.

The Policies pane lets you specify a start and end banner, as well as an error policy which specifies whether to retry or abort the print job, or stop the printer should an error occur. The Access Control pane allows you to deny access to certain users.

The Printer Options pane is where you set particular printing features like paper size and type, print quality, and the input tray to use (see Figure 24-11).

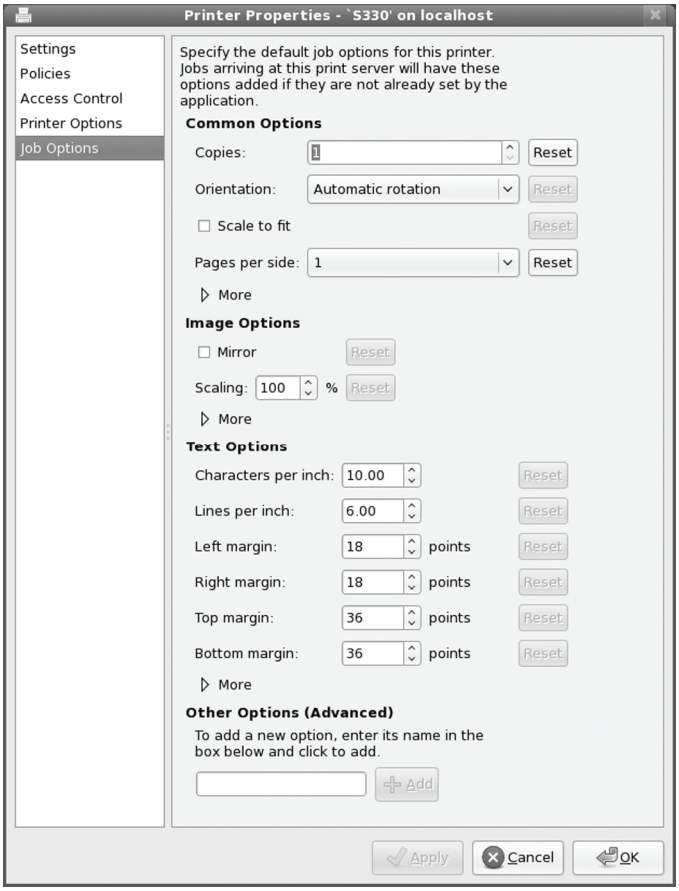


Figure 24-11: Jobs Options pane

On the Job Options pane you can select default printing features (see Figure 24-8). A pop-up menu provides a list of printing feature categories to choose from. You then click the Add button to add the category, selecting a particular feature from a pop-up menu. You can set such features as the number of copies (copies); letter, glossy, or A4-sized paper (media); the kind of document, for instance, text, PDF, PostScript, or image (document format); and single- or double-sided printing (sides).

Default System-wide and Personal Printers

To make printer the default printer, either right-click on the printer icon and select "Set As Default", or single click on the printer icon and then from the Printer configuration window's Printer menu select the "Set As Default" entry (see Figure 24-5 and 24-6). A Set Default Printer dialog open with options for setting the system-wide default or setting the personal default (see Figure 24-12). The system-wide default printer is the default for your entire network served by your CUPS server, not just your local system.

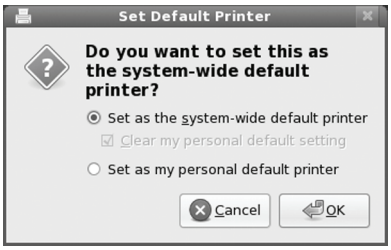


Figure 24-12: Set Default Printer dialog

The system-wide default printer will have a green check mark emblem on its printer icon in the Printer configuration window.

Should you wish to use a different printer yourself as your default printer, you can designate it as your personal default. To make a printer your personal default, select the entry "Set as my personal default printer" in the Set Default Printer dialog. A personal emblem, a heart, will appear on the printer's icon in the Printer configuration window. In Figure 24-13, the S300-windows printer is the system-wide default, whereas the S330 printer is the personal default.

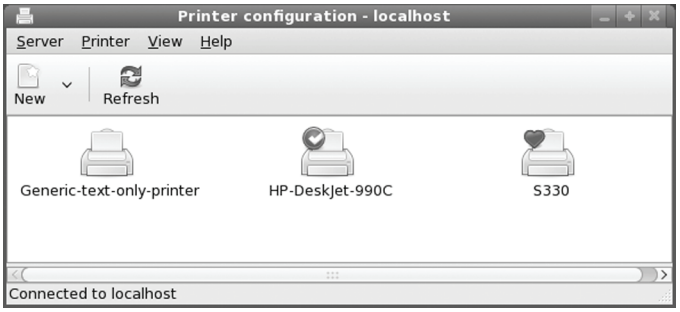


Figure 24-13: System-wide and personal default printers

If you have more than one printer on your system, you can make one the default by clicking Make Default Printer button in the printer's properties Settings pane.

Printer Classes

The Class entry in the New menu lets you create a printer class. You can access the New menu from the Server menu or from the New button. This feature lets you select a group of printers to print a job instead of selecting just one. That way, if one printer is busy or down, another printer can be automatically selected to perform the job. Installed printers can be assigned to different classes. When you click the Class entry in the New menu, a New Class window opens. Here you can enter the name for the class, any comments, and the location (your host name is entered by default). The next screen lists available printers on right side (Other printers) and the printers you assigned to the class on the left side (Printers in this class). Use the arrow button to add or remove printers to the class. Click Apply when finished. The class will appear under the Local Classes heading on the main system-config-printer window. Panes for a selected class are much the same as

for a printer, with a members pane instead of a print control pane. In the Members pane you can change what printers belong to the class

Adding New Printers Manually

Printers are normally detected automatically, though in the case of older printers and network printers, you may need to add the printer manually. In this case click the New button and select Printer. A New Printer window opens up displaying series of dialog boxes where you select the connection, model, drivers, and printer name with location.

On the Select Connection screen, you select the appropriate printer connection information. Connected local printer brands will already be listed by name, such as Canon, whereas for remote printers you specify the type of network connection, like Windows printers via Samba for printers connected to a Windows system, AppSocket/HP Direct for HP printers connected directly to your network. The Internet Printing Protocol (ipp) for printers on Linux and Unix systems on your network.

For most connected printers, your connection is usually determined by the device hotplug services udev and HAL, which now manage all devices. Printers connected to your local system will be first entries on the list. A USB printer will simply be described as a USB printer, using the usb URI designation (see Figure 24-14 and 24-4).

Figure 24-14: Selecting a new printer connection

Figure 24-15: Printer manufacturer for new printers

For an older local printer, you will need to choose the port the printer is connected to, such as LPT1 for the first parallel port used for older parallel printers, or Serial Port #1 for a printer connected to the first serial port. For this example an older parallel printer will be set up, LPT #1.

On the next screen you select your printer manufacturer, choosing it from a printer database (See figure 3-15). You can also choose to load your own printer driver (PPD file).

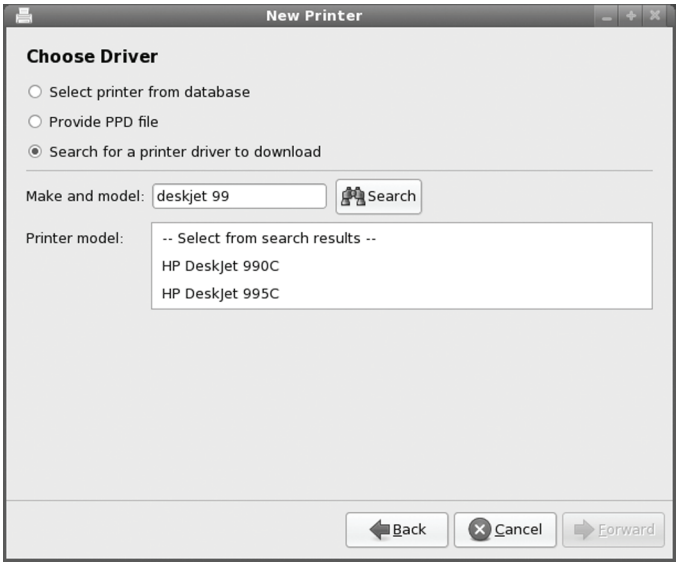


Figure 24-16: Searching for a printer driver from the OpenPrinting repository

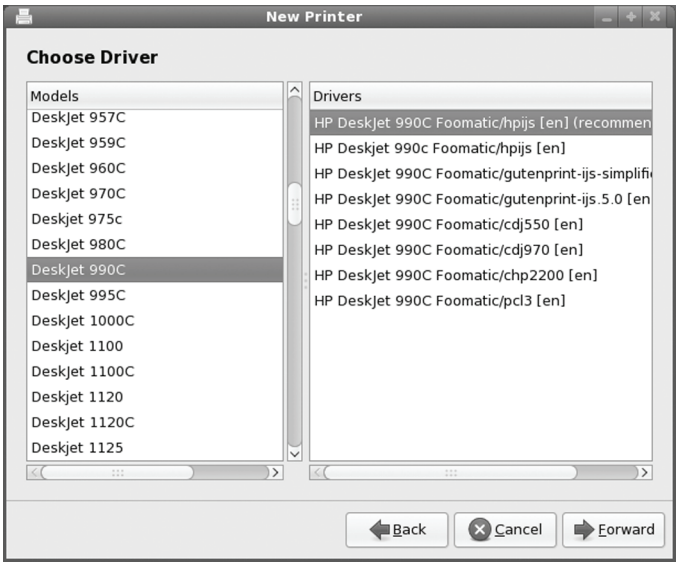


Figure 24-17: Printer Model and driver for new printers using local database

Instead of manually selecting the driver, you can try to search for it on OpenPrinting repository (see Figure 24-16). Click the Search for a printer driver to download button. The Choose Driver window changes to open a "Man and model" text box and a Search button. Enter the printer name and model and click the Search button. You can enter a prefix pattern and matching models will be retrieved. From the pop up menu labeled Printer model, select the driver for your printer. Then click Forward. The found drivers will be listed and you can then select them to download them. If the driver is already available on the local database, then the driver will not be downloaded. Click the Back button and then click "Select driver from database" and choose the driver from there. The Fedora compliant drivers in the Fedora driver database are usually preferable to a downloaded driver.

If you are selecting a printer from the database, then, on the next screen you select that manufacturer's model along with its driver (see Figure 24-17). For some older printer, though the driver can be located on the online repository, you will still choose it from the local database (the drivers are the same). The selected drivers for your printer will be listed. You can find out more about the printer and driver by clicking the Printer and Driver buttons at the bottom of the screen. Then click the Forward button.

You then enter in your printer name and location (see Figure 24-18). These will be entered for you using the printer model and your system's host name. You can change the printer name to anything you want. When ready, click Apply.

Figure 24-18: Printer Name and Location for new printers

You then see an icon for your printer displayed in the Printer configuration window. You are now ready to print.

Configuring Printers with KDE

KDE provides support for adding and configuring CUPS printers through the KDE Control Center. Select the Printers entry under Peripherals. The KDE Printer tool has the capability to perform many different kinds of printing such as sending faxes or saving to PDF files. USB printers that are automatically detected will be listed in the KDE Printer window. When you click on the printer entry, the Information, Jobs, Properties, and Instances panels let you manage your printer and its print jobs. The Properties panel has options for controlling user access, setting quotas, selecting a banner, and even changing your driver.

To change printer options like page size and resolution, you use select the Configure entry from the Printer menu. The Printer menu also lets you disable the printer or test it. The printer toolbar also provides buttons for these commonly performed tasks. The printer manager lets you configure general features like the fonts available, the previewer to use, or the printers to display. A pop-up menu for the printer system used will have CUPS selected by default. You could switch to LPRng if needed. Check the KDEPrint Handbook , accessible from the Documentation menu, for detailed information.

CUPS Web Browser-based configuration tool

One of the easiest way to configure and install printers with CUPS is to use the CUPS configuration tool, which is a web browser–based configuration tool. To start the web interface, enter the following URL into your web browser:

```
http://localhost:631
```

This opens an administration screen where you can manage and add printers. You will first be asked to enter the administrator's username (usually **root**) and password (usually the root user's password).

With the CUPS configuration tool, you install a printer on CUPS through a series of web pages, each of which requests different information. To install a printer, click the Add Printer button to display a page where you enter the printer name and location. The location is the host to which the printer is connected.

Subsequent pages will prompt you to enter the model of the printer and driver, which you select from available listings. Once you have added the printer, you can configure it. Clicking the Manage Printers entry in the Administration page lists your installed printers. You can then click a printer to display a page that lets you control the printer. You can stop the printer, configure its printing, modify its installation, and even delete the printer. Clicking the Configure Printer button displays a page where you can configure how your printer prints, by specifying the resolution or paper size.

Configured information for a printer will be stored in the **/etc/cups/printers.conf** file. You can examine this file directly, even making changes. Here is an example of a printer configuration entry. The **DeviceURI** entry specifies the device used, in this case a USB printer managed by HAL. It is currently idle, with no jobs:

```
# Printer configuration file for CUPS
# Written by cupsd
<Printer mycannon>
Info Cannon s330
Location
DeviceURI
hal:///org/freedesktop/Hal/devices/usb_device_4a9_1074_300HCR_if0_printer_noseria
l
State Idle
StateTime 1166554036
Accepting Yes
Shared Yes
JobSheets none none
QuotaPeriod 0
PageLimit 0
KLimit 0
OpPolicy default
ErrorPolicy stop-printer
</Printer>
```

Configuring Remote Printers on CUPS

To install a remote printer that is attached to a Windows system or another Linux system running CUPS, you specify its location using special URL protocols. For another CUPS printer on a remote host, the protocol used is **ipp**, for Internet Printing Protocol, whereas for a Windows printer, it would be **smb**. Older Unix or Linux systems using LPRng would use the **lpd** protocol.

Be sure your firewall is configured to allow access to remote printers. On the Trusted Services pane in system-config-firewall (System | Administration | Firewall), be sure that the Samba and IPP services are checked. Samba allows access for Windows printers, and IPP allows

access for Internet Printer Protocol printers usually found on other Linux systems. There will be entries for the Samba client and server, as well as IPP client and server (see Figure 24-19).

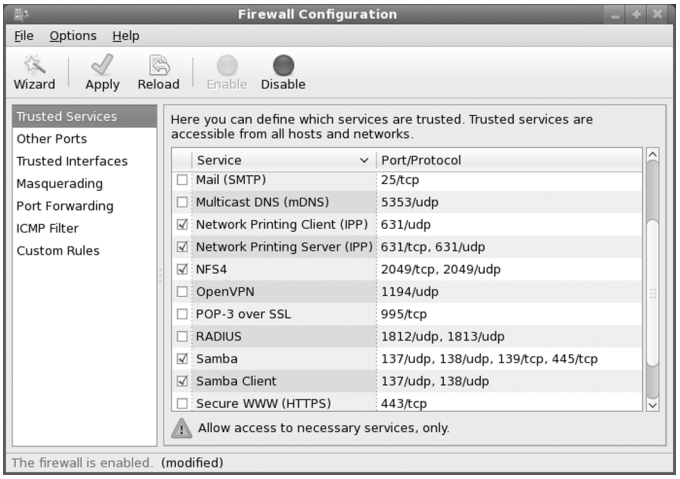


Figure 24-19: Firewall configuration for a remote printer

To access an SMB shared remote printer, you need to install Samba and have the Server Message Block services enabled using the smb and nmb daemons. The Samba service will be enabled by default. The service is enabled by checking the Windows Folders entry in the Gnome Services tool (System | Administration | Services). Printer sharing must, in turn, be enabled on the Windows network.

Configuring Remote Printers with system-config-printers

To install a remote printer that is attached to a Windows system or another Linux system running CUPS, you specify its location using special URI protocols. For a locally attached USB printer, the USB URI is uses, **usb**. For another CUPS printer on a remote host, the protocol used is **ipp**, for Internet Printing Protocol, whereas for a Windows printer, it would be **smb**. Older UNIX or Linux systems using LPRng would use the **lpd** protocol

You can also use system-config-printer to set up a remote printer on Linux, UNIX, or Windows networks. When you add a new printer or edit one, the New Printer/Select Connection dialog will list possible remote connection types. When you select a remote connection entry, a pane will be displayed where you can enter configuration information.

For a remote Linux or UNIX printer, select either Internet Printing Protocol (ipp), which is used for newer systems, or LPD/LPR Host or Printer, which is used for older systems. Both panes display entries for the Host name and the Printer name. For the Host name, enter the hostname for the system that controls the printer. For the Printer name, enter the device name on that host for the printer. The LPD/LPR dialog also has a probe button for detecting the printer.

For an Apple or HP jet direct printer on your network, select the AppSocket/HP jetDirect entry. You are prompted to enter the IP address and printer name.

Figure 24-20: Selecting a Windows printer

A "Windows printer via Samba" is one located on a Windows network (see Figure 24-20). You need to specify the Windows server (host name or IP address), the name of the share, the name of the printer's workgroup, and the username and password. The format of the printer SMB URL is shown on the SMP Printer pane. The share is the hostname and printer name in the **smb** URI format *//workgroup/hostname/printername.* The workgroup is the windows network workgroup that the printer belongs to. On small networks there is usually only one. The hostname is the computer where the printer is located. The username and password can be for the printer resource itself, or for access by a particular user. The pane will display a box at the top where you can enter the share host and printer name as a **smb** URI.

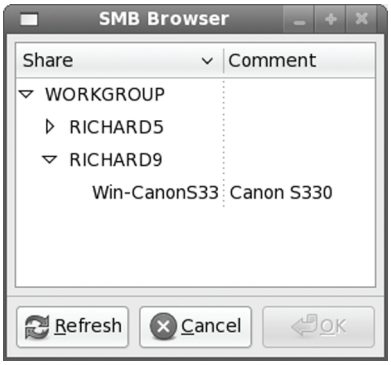


Figure 24-21: SMB Browser, selecting a remote windows printer

You can click the Browse button to open a SMB Browser window, where you can select the printer from a listing of Windows hosts on your network (see Figure 24-21). For example, if your Windows network is WORKGROUP, then the entry WORKGROUP will be shown, which

you can then expand to list all the Windows hosts on that network (if your network is MSHOME, then that is what will be listed).

When you make your selection, the corresponding URL will show up in the **smb://** box (See Figure 24-22). If you are using the Firestarter firewall, be sure to turn it off before browsing a Windows workgroup for a printer, unless already configured to allow Samba access. Also on the pane, you can enter in any needed Samba authentication, if required, like user name or password. Check "Authentication required" to allow you to enter the Samba Username and Password.

Figure 24-22: Remote windows printer connection configuration

You then continue with install screens for the printer model, driver, and name. Once installed, you can then access the printer properties just as you would any printer.

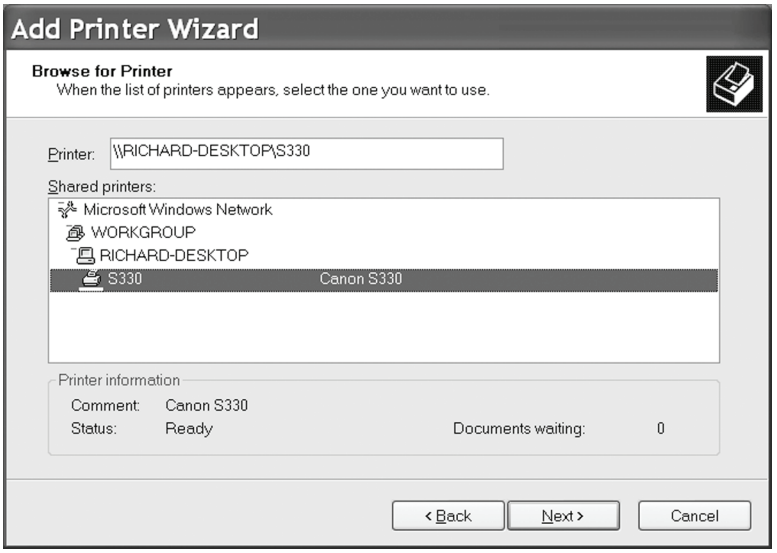


Figure 24-23: Remote Linux printer selection on Windows

Linux Printers remotely accessed from Windows

On a Windows system, like WIndows XP, you can use the Add Printer Wizard to locate a shared printer on a Linux system (see Figure 10). Locate the Fedora system, click on it, and the shared printers on the Fedora system will be listed.

Tip: If the Windows driver for your printer on your Windows system should fail, you could just attach printer to a Linux system and use the Linux drivers, even remotely accessing the printer from your Windows system.

Configuring remote printers manually

In the **cupsd.conf** file, for a remote printer, the DeviceURI entry, instead of listing the device, will have an Internet address, along with its protocol. For example, a remote printer on a CUPS server (**ipp**) would be indicated as shown here (a Windows printer would use an **smb** protocol):

```
DeviceURI ipp://mytsuff.com/printers/queue1
```

For a Windows printer, you first need to install, configure, and run Samba. (CUPS uses Samba to access Windows printers.) When you install the Windows printer on CUPS, you specify its location using the URL protocol **smb**. The user allowed to log in to the printer is entered before the hostname and separated from it by an `@` sign. On most configurations, this is the **guest** user. The location entry for a Windows printer called **myhp** attached to a Windows host named **lizard** is shown here. Its Samba share reference would be **//lizard/myhp**:

```
DeviceURI smb://guest@lizard/myhp
```

To enable CUPS on Samba, you also have to set the printing option in the **/etc/samba/smb.conf** file to **`cups`**, as shown here:

```
printing = cups
printcap name = cups
```

To enable CUPS to work with Samba, you have to link the **smbspool** to the CUPS **smb** spool directory:

```
ln -s /usr/bin/smbspool   /usr/cups/backend/smb
```

Note: To configure a shared Linux printer for access by Windows hosts, you need to configure it as a SMB shared printer. You do this with Samba.

CUPS Web Configuration Interface

To start the Web interface, enter the following URL into your Web browser:

```
http://localhost:631
```

This opens the Home screen for the CUPS Web interface. There are tabs for various sections, as well as buttons for specialized tasks like adding printers (see Figure 24-24). Tabs including Administration, Classes, Documentation, Jobs, and Printers. You can manage and add printers. The Printers tab will list installed printers with buttons for accessing their print queues,

printer options, and job options, among others. The Jobs tab lists your print jobs and lets you manage them.

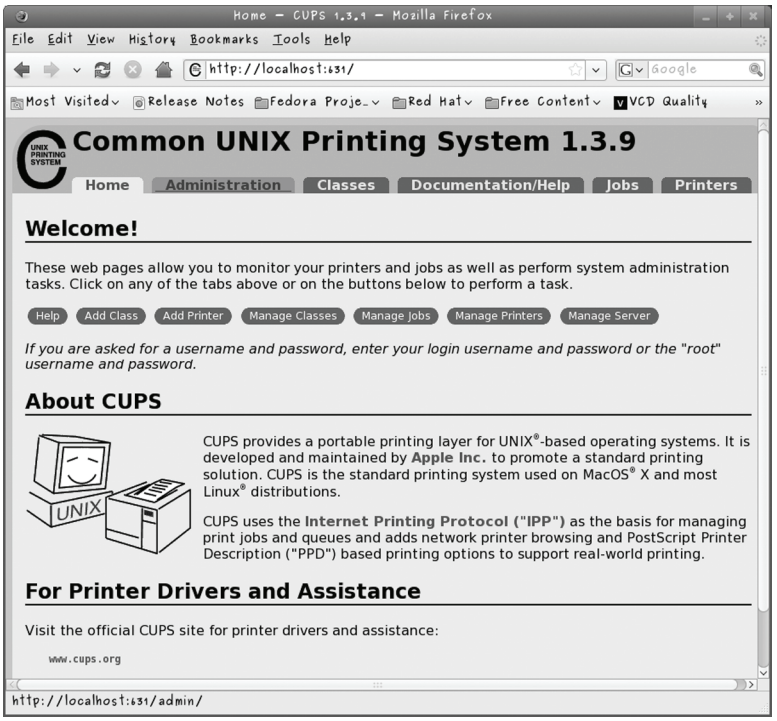


Figure 24-24: CUPS Web-based Configuration Tool

When you try to make any changes for the first time during the session, you will first be asked to enter the administrator's username (**root**) and password (the root user's password).

The Administration tab displays segments for Printers, Classes, Jobs, and the Server (see Figure 24-25). The Sever section is where you allow printer sharing. Buttons allow you to view logs and change settings.

With the CUPS configuration tool, you install a printer on CUPS through a series of Web pages, each of which requests different information. To install a printer, click the Add Printer button either on the Home page or the Administration page. This displays a page where you enter the printer name and location (see Figure 24-26). The location is the host to which the printer is connected. The procedure is similar to **system-config-printer**. Subsequent pages will prompt you to enter the device, make and model of the printer, and the driver, which you select from available listings.

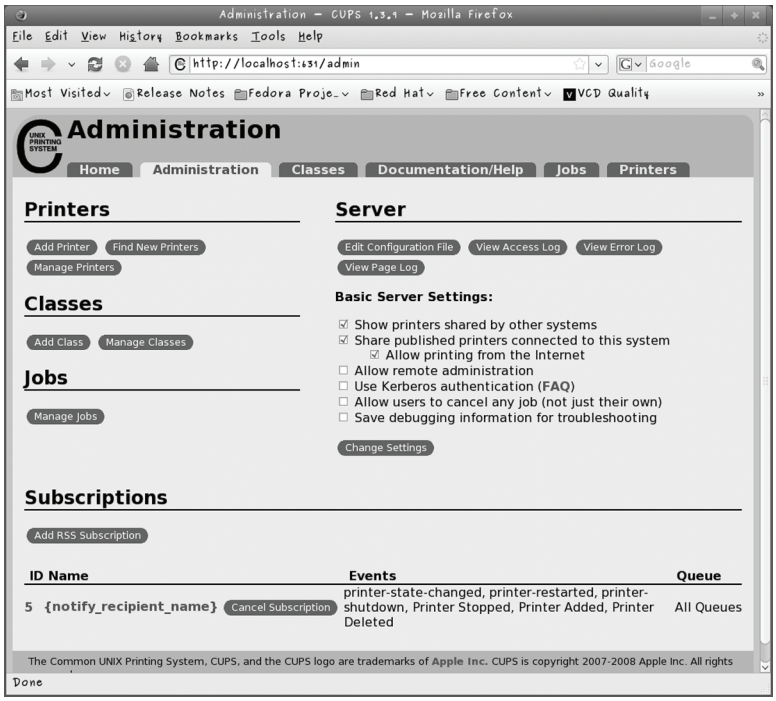


Figure 24-25: CUPS Web-based Administration tab

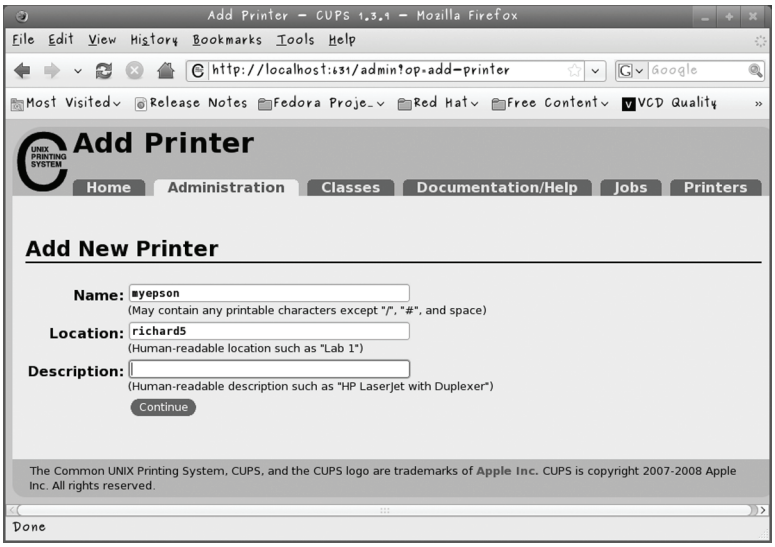


Figure 24-26: Adding a new printer: CUP Web interface

.Once you have added the printer, you can configure it. Clicking the Printers tab or the Manage Printers button in the Administration page. The Printers page will list your installed

printers (see Figure 24-27). For each printer, a list of buttons lets you perform certain tasks, including button. You can stop the printer, configure its printing, modify its installation, and even delete the printer. Clicking the Set Printer Options button displays a page on the Administration tab where you can configure how your printer prints, by specifying the resolution or paper size.

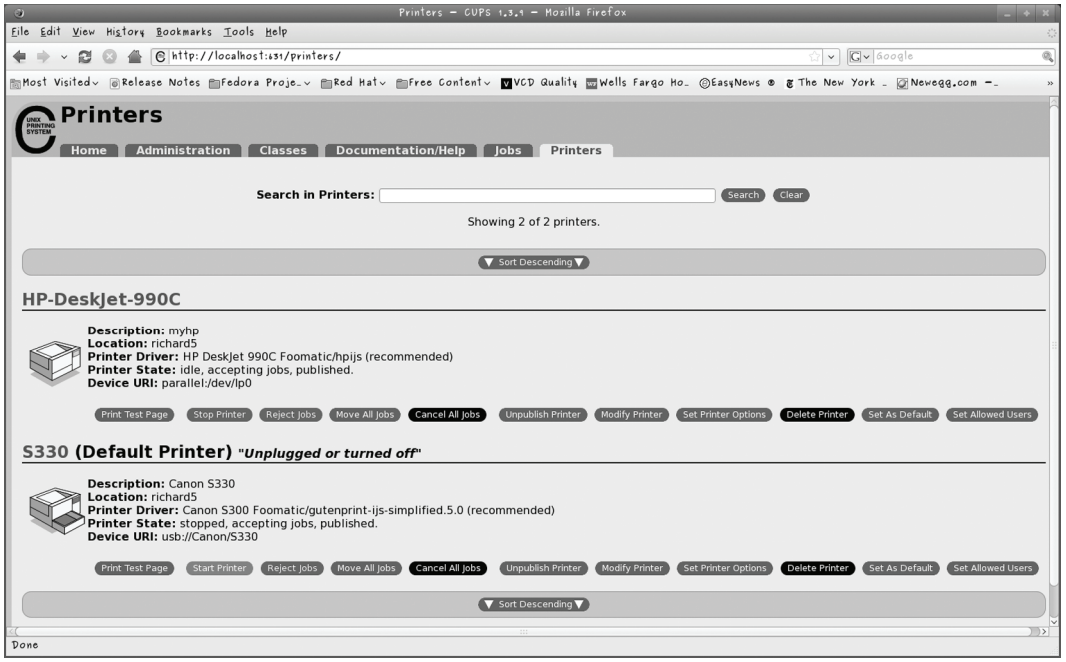


Figure 24-27: CUPS Printers: CUP Web interface

CUPS Configuration files

CUPS configuration files are placed in the **/etc/cups** directory. These files are listed in Table 24-2. The **classes.conf**, **printers.conf**, and **client.conf** files can be managed by the web interface. The **printers.conf** file contains the configuration information for the different printers you have installed. Any of these files can be edited manually, if you wish.

Filename	Description
classes.conf	Contains configurations for different local printer classes
client.conf	Lists specific options for specified clients
cupsd.conf	Configures the CUPS server, **cupsd**
printers.conf	Contains printer configurations for available local printers

Table 24-2: CUPS Configuration Files

cupsd.conf and printers.conf

The CUPS server is configured with the **cupsd.conf** file located in **/etc/cups**. You must edit configuration options manually; the server is not configured with the web interface. Your installation of CUPS installs a commented version of the **cupsd.conf** file with each option listed, though most options will be commented out. Commented lines are preceded with a **#** symbol. Each option is documented in detail. The server configuration uses an Apache web server syntax consisting of a set of directives. As with Apache, several of these directives can group other directives into blocks.

Configured information for a printer will be stored in the **/etc/cups/printers.conf** file. You can examine this file directly, even making changes. Here is an example of a printer configuration entry. Notice that it was created using system-config-printer. The **DeviceURI** entry specifies the device used, in this case a USB printer managed by HAL. It is currently idle, with no jobs:

```
# Printer configuration file for CUPS
# Written by cupsd
<DefaultPrinter s330>
Info Cannon s330
Location
DeviceURI usb://Canon/S330
hal:///org/freedesktop/Hal/devices/usb_device_4a9_1074_300HCR_if0_printer_noseria
l
State Idle
StateTime 1166554036
Accepting Yes
Shared Yes
JobSheets none none
QuotaPeriod 0
PageLimit 0
KLimit 0
OpPolicy default
ErrorPolicy stop-printer
</Printer
```

CUPS Directives

Certain directives allow you to place access controls on specific locations. These can be printers or resources, such as the administrative tool or the spool directories. Location controls are implemented with the **Location** directive. **Allow From** and **Deny From** directives can permit or deny access from specific hosts. CUPS supports both Basic and Digest forms of authentication, specified in the **AuthType** directive. Basic authentication uses a user and password. For example, to use the web interface, you are prompted to enter the root user and the root user password. Digest authentication makes use of user and password information kept in the CUPS **/etc/cups/passwd.md5** file, using MD5 versions of a user and password for authentication. The **AuthClass** directive specifies the class allowed access. The **System** class includes the root, sys, and system users. The following example shows the **Location** directive for the **/admin** resource, the administrative tool:

```
<Location /admin>

AuthType Basic
AuthClass System

## Restrict access to local domain
Order Deny,Allow
Deny From All
Allow From 127.0.0.1

</Location>
```

CUPS Command Line Print Clients

Once a print job is placed on a print queue, you can use any of several print clients to manage the printing jobs on your printer or printers, such as Klpq, the GNOME Print Manager, and the CUPS Printer Configuration tool for CUPS. You can also use several command line print CUPS clients. These include the **lpr**, **lpc**, **lpq**, and **lprm** commands. With these clients, you can print documents, list a print queue, reorder it, and remove print jobs, effectively canceling them. For network connections, CUPS features an encryption option for its commands, **-E**, to encrypt print jobs and print information sent of a network. Table 24-3 shows various printer commands.

Note: The command line clients have the same name, and much the same syntax, as the older LPR and LPRng command line clients used in Unix and older Linux systems.

Printer Management	Description
GNOME Print Manager	GNOME print queue management tool (CUPS).
CUPS Configuration Tool	Prints, manages, and configures CUPS.
lpr *options file-list*	Prints a file, copies the file to the printer's spool directory, and places it on the print queue to be printed in turn. **-P** *printer* prints the file on the specified printer.
lpq *options*	Displays the print jobs in the print queue. **-P** *printer* prints the queue for the specified printer. **-l** prints a detailed listing.
lpstat *options*	Displays printer status.
lprm *options printjob-id* **or** *printer*	Removes a print job from the print queue. You identify a particular print job by its number as listed by **lpq**. **-P** *printer* removes all print jobs for the specified printer.
lpc	Manages your printers. At the **lpc>** prompt, you can enter commands to check the status of your printers and take other actions.

Table 24-3: CUPS Print Clients

lpr

The **lpr** client submits a job, and **lpd** then takes it in turn and places it on the appropriate print queue; **lpr** takes as its argument the name of a file. If no printer is specified, then the default printer is used. The **-P** option enables you to specify a particular printer. In the next example, the user first prints the file **preface** and then prints the file **report** to the printer with the name **myepson**:

```
$ lpr preface
$ lpr -P myepson report
```

lpc

You can use **lpc** to enable or disable printers, reorder their print queues, and re-execute configuration files. To use **lpc**, enter the command **lpc** at the shell prompt. You are then given an **lpc>** prompt at which you can enter **lpc** commands to manage your printers and reorder their jobs. The **status** command with the name of the printer displays whether the printer is ready, how many print jobs it has, and so on. The **stop** and **start** commands can stop a printer and start it back up. The printers shown depend on the printers configured for a particular print servers. A printer configured on CUPS will only show if you have switched to CUPS.

```
# lpc
lpc> status myepson
myepson:
 printer is on device 'hal' speed -1
 queuing is enabled
 printing is enabled
 1 entry in spool area
```

lpq and lpstat

You can manage the print queue using the **lpq** and **lprm** commands. The **lpq** command lists the printing jobs currently on the print queue. With the **-P** option and the printer name, you can list the jobs for a particular printer. If you specify a username, you can list the print jobs for that user. With the **-l** option, **lpq** displays detailed information about each job. If you want information on a specific job, simply use that job's ID number with **lpq**. To check the status of a printer, use **lpstat**.

```
# lpq
myepson is ready and printing
Rank    Owner  Jobs  File(s)          Total Size
active  chris    1   report           1024
```

lprm

The **lprm** command enables you to remove a print job from the queue, erasing the job before it can be printed. The **lprm** command takes many of the same options as **lpq**. To remove a specific job, use **lprm** with the job number. To remove all printing jobs for a particular printer, use the **-P** option with the printer name. **lprm** with no options removes the job printing currently. The following command removes the first print job in the queue (use **lpq** to obtain the job number):

```
# lprm 1
```

CUPS Command Line Administrative Tools

CUPS provides command line administrative tools like **lpadmin**, **lpoptions**, **lpinfo**, **enable**, **disable**, **accept**, and **reject**. The **enable** and **disable** commands start and stop print queues directly, whereas the **accept** and **reject** commands start and stop particular jobs. The **lpinfo** command provides information about printers, and **lpoptions** lets you set printing options. The **lpadmin** command lets you perform administrative tasks like adding printers and changing configurations. CUPS administrative tools are listed in Table 24-4.

lpadmin

You can use the **lpadmin** command to either set the default printer or configure various options for a printer. You can use the **-d** option to specify a particular printer as the default destination. Here **myepson** is made the default printer:

```
lpadmin -d myepson
```

The **-p** option lets you designate a printer for which to set various options. The following example sets printer description information:

```
lpadmin -p myepson  -D  Epson550
```

Certain options let you control per-user quotas for print jobs. The **job-k-limit** option sets the size of a job allowed per user, **job-page-limit** sets the page limit for a job, and **job-quota-period** limits the number of jobs with a specified time frame. The following command set a page limit of 100 for each user:

```
lpadmin -p myepson  -o job-page-limit=100
```

User access control is determined with the **-u** option with an **allow** or **deny** list. Users allowed access are listed following the **allow**: entry, and those denied access are listed with a **deny:** entry. Here access is granted to **chris** but denied to **aleina** and **larisa**.

```
lpadmin -p myepson -u allow:chris  deny:aleina,larisa
```

Administration Tool	Description
lpadmin	CUPS printer configuration
lpoptions	Sets printing options
enable	Activates a printer
disable	Stops a printer
accept	Allows a printer to accept new jobs
reject	Prevents a printer from accepting print jobs
lpinfo	Lists CUPS devices available

Table 24-4: CUPS Administrative Tools

Use **all** or **none** to permit or deny access to all or no users. You can create exceptions by using **all** or **none** in combination with user-specific access. The following example allows access to all users except **justin**:

```
lpadmin -p myepson  -u allow:all   deny:justin
```

lpoptions

The **lpoptions** command lets you set printing options and defaults that mostly govern how your print jobs will be printed. For example, you can set the color or page format to be used with a particular printer. Default settings for all users are maintained by the root user in the **/etc/cups/lpoptions** file, and each user can create their own configurations, which are saved in their **.lpoptions** files. The **-l** option lists current options for a printer, and the **-p** option designates a printer (you can also set the default printer to use with the **-d** option).

```
lpoptions -p myepson -l
```

Printer options are set using the **-o** option along with the option name and value, **-o** *option*=*value*. You can remove a printer option with the **-r** option. For example, to print on both sides of your sheets, you can set the **sides** option to **two-sided**:

```
lpoptions -p myepson -o sides=two-sided
```

To remove the option, use **-r**.

```
lpoptions -p myepson -r sides
```

To display a listing of available options, check the standard printing options in the CUPS Software Manual at **www.cups.org**.

enable and disable

The **enable** command starts a printer, and the **disable** command stops it. With the **-c** option, you can cancel all jobs on the printer's queue, and the **-r** option broadcasts a message explaining the shutdown.

```
disable myepson
```

accept and reject

The **accept** and **reject** commands let you control access to the printer queues for specific printers. The **reject** command prevents a printer from accepting jobs, whereas **accept** allows new print jobs.

```
reject myepson
```

lpinfo

The **lpinfo** command is a handy tool for letting you know what CUPS devices and drivers that are available on your system. Use the **-v** option for devices and the **-m** option for drivers.

```
lpinfo -m
```

fedora

25. Network File Systems and Network Information Service: NFS and NIS

Linux provides several tools for accessing files on remote systems connected to a network. The Network File System (NFS) enables you to connect to and directly access resources such as files or devices like CD-ROMs that reside on another machine. The new version, NFS4, provides greater security, with access allowed by your firewall. The Network Information Service (NIS) maintains configuration files for all systems on a network.

Portmapper and RPC Services

Both NFS and NIS are Remote Procedure Call services (RPC). To work, you first have to have the portmappter service enabled and running. The portmapper maps an address to an RPC service, locating the RPC service. It tells hosts where the NFS and NIS services can be found on the system. The name of the portmapper service is **rpcbind**. Use system-config-services or the service command to start it if it not already running. You can use **chkconfig** to have it automatically start.

```
service rpcbind start
```

Network File System: NFS

NFS enables you to mount a file system on a remote computer as if it were local to your own system. You can then directly access any of the files on that remote file system. This has the advantage of allowing different systems on a network to access the same files directly, without each having to keep its own copy. Only one copy will be on a remote file system, which each computer can then access. You can find out more about NFS at its website at **http://nfs.sourceforge.net**.

NFS Daemons

NFS operates over a TCP/IP network. The remote computer that holds the file system makes it available to other computers on the network. It does so by exporting the file system, which entails making entries in an NFS configuration file called **/etc/exports**, as well as by running several daemons to support access by other systems. These include **rpc.mountd**, **rpc.nfsd**, and **rpc.portmapper**. Access to your NFS server can be controlled by the **/etc/hosts.allow** and **/etc/hosts.deny** files. The NFS daemons are listed here:

- **rpc.nfsd** Receives NFS requests from remote systems and translates them into requests for the local system.
- **rpc.mountd** Performs requested mount and unmount operations.
- **rpc.rquotad** Provides user disk quota management.
- **rpc.statd** Provides locking services when a remote host reboots.
- **rpc.lockd** Handles lock recovery for systems that have gone down.
- **rpc.idmapd** Maps user and group IDs to names.
- **rpc.gssd** Client support for the rpcsec_gss protocol for gss-api security in NFSv4.
- **rpc.svcgssd** Server support for the rpcsec_gss protocol for gss-api security in NFSv4.

You can start up and shut down the NFS daemons using the **/etc/rc.d/init.d/nfs** service, which you can invoke with the **service** command, **./sbin/service nfs start**. To have NFS started automatically, you can use **chkconfig** or the **system-config-services** tool to specify the runlevels at which it will operate. The following example will have NFS start up automatically at runlevels 3 and 5. The NFS service should already be configured to start up automatically at runlevels 3 and 5. You can use **system-config-services** to check, (System | Administration | Services), select the nfs entry.

```
chkconfig -level 35  nfs on
```

The **nfs** service script will start up the **nfsd**, **mountd**, **idmapd**, **svcgssd**, and **rquotad** daemons. To enable NFS locking, you use the **nfslock** script. This will start up the **statd** and **lockd** daemons. NFS locking provides for better recovery from interrupted operations that can occur from system crashes on remote hosts.

The Fedora NFS server and support tools are located in the following packages.

nfs-utils	NFS server and supporting tools
system-config-nfs	Fedora NFS desktop configuration tool
nfs4-acl-tools	Access control list commands for NFS4 supported security
nfswatch	Monitor NFS usage

Make sure that your firewall allows NFS access. On the system-config-firewall Trusted Services panel, make sure that NFS4 is checked.

Also be sure to allow SELinux to permit NFS access. Using system-config-selinux (System | Administration | Selinux Management) select Boolean on the sidebar, and then locate NFS and make sure "Support NFS home directories" is checked.

Setting up NFS Directories with system-config-nfs: Shared Folders

You can set up and NFS shared folder easily using the **system-config-nfs** tool on the Fedora desktop. Select NFS Server from the System | Administration | Server Settings menu. This opens the NFS Server Configuration Tool window as shown in Figure 25-1. Be sure your NFS server software is installed and that the NFS server is running (system-config-services).

Note: It is advisable to use NFS on a local secure network only. If used over the Internet, NFS opens your system up to nonsecure access.

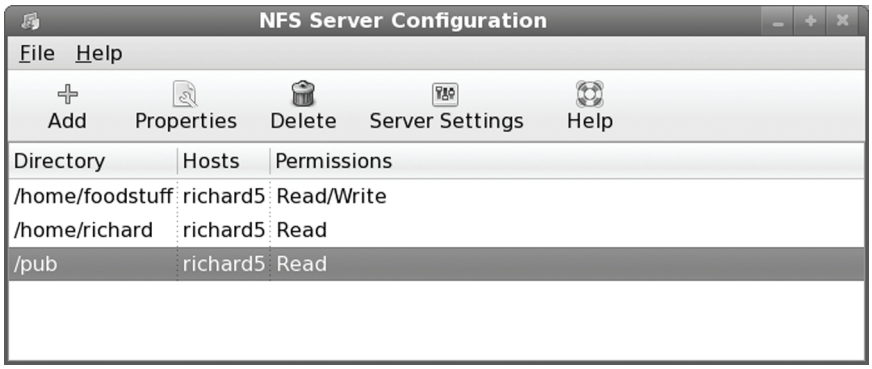


Figure 25-1: system-config-nfs

To add a directory to share, click the Add button to open an Add NFS Share dialog window, similar to the Edit NFS Share window shown in Figure 25-2. You can browse for or enter the directory to be shared in the Directory box. You then list the host where it is located, and specify the access permissions.

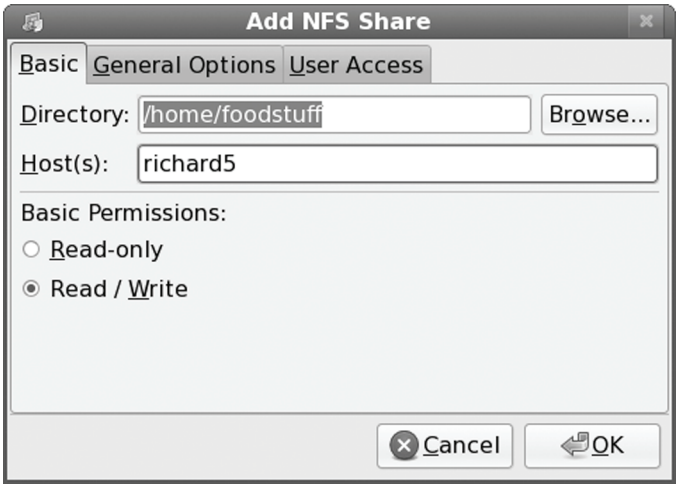


Figure 25-2: system-config-nfs basic share properties

On the General Options tab, you can set basic security and update features such as insecure file locking or immediate syncing of write operations. By default, the Sync Write Operations On Request option is set (see Figure 25-3).

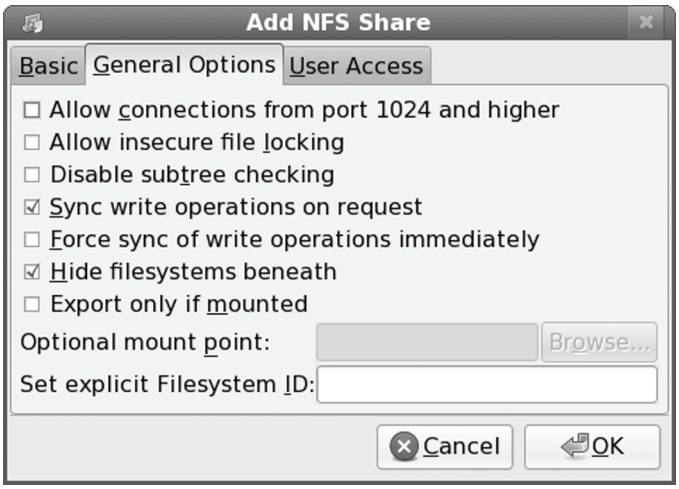


Figure 25-3: system-config-nfs general options

On the User Access tab, you can treat users as local root users or as anonymous users with specific user or group IDs (see Figure 25-4).

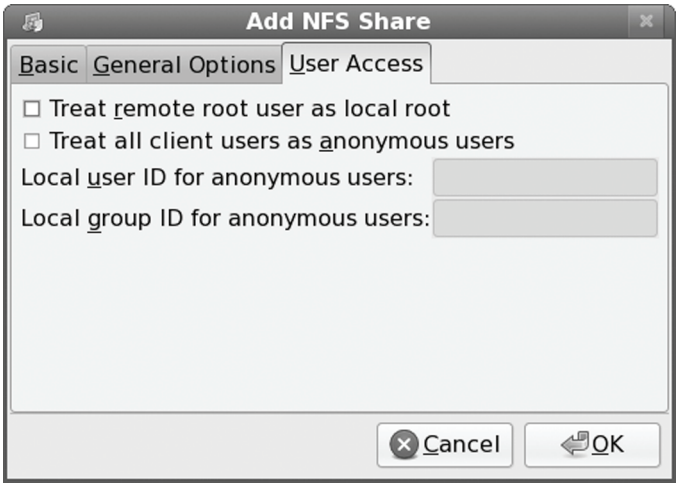


Figure 25-4: system-config-nfs User Access options

When you click OK, the entry will be listed in the NFS Server Configuration window. Changes do take effect as soon as you click OK. This will create an entry in the **/etc/exports** file for the shared directory. You can later change any of these settings by selecting the entry and clicking Properties to open an Edit dialog box. To save your settings, click Apply.

Figure 25-1 shows the same entries that are used in the **/etc/exports** file described in the next section. The /etc/exports file generated by the NFS Server Configuration tool using this example is shown here:

```
/home/foodstuff       richard5(rw,sync)
/home/richard         richard5(ro,sync)
/pub                  richard5(ro,sync,no_wdelay,insecure_locks,all_squash)
```

General Option	Description
secure	Requires that requests originate on secure ports, those less than 1024. This is on by default.
insecure	Turns off the **secure** option.
ro	Allows only read-only access. This is the default.
rw	Allows read/write access.
sync	Performs all writes when requested. This is the default.
async	Performs all writes when the server is ready.
no_wdelay	Performs writes immediately, not checking to see if they are related.
wdelay	Checks to see if writes are related, and if so, waits to perform them together. Can degrade performance. This is the default.
hide	Automatically hides an exported directory that is the subdirectory of another exported directory.
subtree_check	Checks parent directories in a file system to validate an exported subdirectory. This is the default.
no_subtree_check	Does not check parent directories in a file system to validate an exported subdirectory.
insecure_locks	Does not require authentication of locking requests. Used for older NFS versions.
User ID Mapping	**Description**
all_squash	Maps all UIDs and GIDs to the anonymous user. Useful for NFS-exported public FTP directories, news spool directories, and so forth.
no_all_squash	The opposite option to **all_squash**. This is the default setting.
root_squash	Maps requests from remote root user to the anonymous UID/GID. This is the default.
no_root_squash	Turns off root squashing. Allows the root user to access as the remote root.
anonuid	Sets explicitly the UID and GID of the anonymous account used for **all_squash** and **root_squash** options. The defaults are nobody and nogroup.

Table 25-1: The /etc/exports Options

Options set in the Basic, General Options, and User Access panels show up as options listed for each entry. For example, Read access is ro and Read/Write access is rw. Check the NFS Server Configuration help documents for a complete listing. Options are listed in Table 25-1.

NFS Configuration: /etc/exports

An entry in the **/etc/exports** file specifies the file system to be exported and the hosts on the network that can access it. For the file system, enter its *mountpoint,* the directory to which it was mounted on the host system. This is followed by a list of hosts that can access this file system along with options to control that access. A comma-separated list of export options placed within a set of parentheses may follow each host. For example, you might want to give one host read-only access and another read and write access. If the options are preceded by an ***** symbol, they are applied to any host. A list of options is provided in Table 25-1. The format of an entry in the **/etc/exports** file is shown here:

```
directory-pathname   host-designation(options)
```

NFS Host Entries

You can have several host entries for the same directory, each with access to that directory:

```
directory-pathname   host(options) host(options)  host(options)
```

You have a great deal of flexibility when specifying hosts. For hosts within your domain, you can just use the hostname, whereas for those outside, you need to use a fully qualified domain name. You can also use just the host's IP address. Instead of a single host, you can reference all the hosts within a specific domain, allowing access by an entire network. A simple way to do this is to use the ***** for the host segment, followed by the domain name for the network, such as ***.mytrek.com** for all the hosts in the **mytrek.com** network. Instead of domain names, you can use IP network addresses with a CNDR format where you specify the netmask to indicate a range of IP addresses. You can also use an NIS netgroup name to reference a collection of hosts. The NIS netgroup name is preceded by an **@** sign.

```
directory      host(options)
directory      *(options)
directory      *.domain(options)
directory      192.168.1.0/255.255.255.0(options)
directory      @netgroup(options)
```

NFS Options

Options in **/etc/exports** operate as permissions to control access to exported directories. Read-only access is set with the **ro** option, and read/write with the **rw** option. The **sync** and **async** options specify whether a write operation is performed immediately (**sync**) or when the server is ready to handle it (**async**). By default, write requests are checked to see if they are related, and if so, they are written together (**wdelay**). This can degrade performance. You can override this default with **no_wdelay** and have writes executed as they are requested. If two directories are exported, where one is the subdirectory of another, the subdirectory is not accessible unless it is explicitly mounted (**hide**). In other words, mounting the parent directory does not make the subdirectory accessible. The subdirectory remains hidden until also mounted. You can overcome this restriction with the **no_hide** option (though this can cause problems with some file systems). If an exported directory is actually a subdirectory in a larger file system, its parent directories are checked to make sure that the subdirectory is the valid directory (**subtree_check**). This option works well with read-only file systems but can cause problems for write-enabled file systems,

where filenames and directories can be changed. You can cancel this check with the **no_subtree_check** option.

NFS User-Level Access

Along with general options, there are also options that apply to user-level access. As a security measure, the client's root user is treated as an anonymous user by the NFS server. This is known as *squashing* the user. In the case of the client root user, squashing prevents the client from attempting to appear as the NFS server's root user. Should you want a particular client's root user to have root-level control over the NFS server, you can specify the **no_root_squash** option. To prevent any client user from attempting to appear as a user on the NFS server, you can classify them as anonymous users (the **all_squash** option). Such anonymous users only have access to directories and files that are part of the anonymous group.

Normally, if a user on a client system has a user account on the NFS server, that user can mount and access his or her files on the NFS server. However, NFS requires the User ID for the user be the same on both systems. If this is not the case, he or she is considered to be two different users. To overcome this problem, you can use an NIS service, maintaining User ID information in just one place, the NIS password file.

NFS /etc/exports Example

Examples of entries in an **/etc/exports** file are shown here. Read-only access is given to all hosts to the file system mounted on the **/pub** directory, a common name used for public access. Users, however, are treated as anonymous users (**all_squash**). Read and write access is given to the **lizard.mytrek.com** computer for the file system mounted on the **/home/mypics** directory. The next entry allows access by **rabbit.mytrek.com** to the NFS server's CD-ROM, using only read access. The last entry allows anyone secure access to **/home/richlp**.

/etc/exports

```
/pub                *(ro,insecure,all_squash,sync)
/home/mypics       lizard.mytrek.com(rw,sync)
/media/cdrom        rabbit.mytrek.com(ro,sync)
/home/richlp        *(secure,sync)
```

Applying Changes

Each time your system starts up the NFS server (usually when the system starts up), the **/etc/exports** file will be read and those directories specified will be exported. When a directory is exported, an entry for it is made in the **/var/lib/nfs/xtab** file. It is this file that NFS reads and uses to perform the actual exports. Entries are read from **/etc/exports** and corresponding entries made in **/var/lib/nfs/xtab**. The **xtab** file maintains the list of actual exports.

If you want to export added entries in the **/etc/exports** file immediately, without rebooting, you can use the **exportfs** command with the **-a** option. It is helpful to add the **-v** option to display the actions that NFS is taking. Use the same options to effect any changes you make to the **/etc/exports** file.

```
exportfs -a -v
```

If you later make changes to the **/etc/exports** file, you can use the **-r** option to re-export its entries. The **-r** option will re-sync the **/var/lib/nfs/xtab** file with the **/etc/exports** entries, removing any other exports or any with different options.

```
exportfs -r -v
```

To both export added entries and re-export changed ones, you can combine the **-r** and **-a** options.

```
exportfs -r -a -v
```

Manually Exporting File Systems

You can also use the **exportfs** command to manually export file systems instead of using entries for them in the **/etc/exports** file. Export entries will be added to the **/var/lib/nfs/xtab** file directly. With the **-o** option, you can list various permissions and then follow them with the host and file system to export. The host and file system are separated by a colon. For example, to manually export the **/home/myprojects** directory to **golf.mytrek.com** with the permissions **ro** and **insecure**, you use the following:

```
exportfs -o rw,insecure golf.mytrek.com:/home/myprojects
```

You can also use **exportfs** to un-export a directory that has already been exported, either manually or by the **/etc/exports** file. Just use the **-u** option with the host and the directory exported. The entry for the export will be removed from the **/var/lib/nfs/xtab** file. The following example will un-export the **/home/foodstuff** directory that was exported to **lizard.mytrek.com**:

```
exportfs -u lizard.mytrek.com:/home/foodstuff
```

NFSv4

NFS version 4 is a new version of the NFS protocol with enhanced features like greater security, reliability, and speed. Most of the commands are the same with a few changes. For example, when you mount an NFSv4 file system, you need to specify the **nfs4** file type. Also, for NFSv4, in the **/etc/exports** file, you can use the **fsid=0** option to specify the root export location.

```
 /home/richlp          *(fsid=0,ro,sync)
```

The preceding entry lets you mount the file system to the **/home/richlp** directory without having to specify it in the mount operation.

```
mount -t nfs4  rabbit.mytrek.com:/  /home/dylan/projects
```

NFSv4 also supports the RPCSEC_GSS (Remote Procedure Call Security, Generic Security Services) security mechanism which provides for private/public keys, encryption, and authentication with support for Kerberos.

Tip: To see what file systems are exported currently by an NFS server, you can use the **showmount** command with the **-e** option, **showmount -e** *nfserver*. To see what client hosts mount file systems on the NFS server, you use **showmount** with no options.

NFS File and Directory Security with NFS4 Access Lists

With NFS4 you can set up access control lists (ACL) for particular directories and files. You use the NFS4 ACL tools to manage these lists (**nfs4-acl-tools** package). The NFS4 file system ACL tools include **nfs4_getfacl**, **nfs4_setfacl**, and **nfs4_editfacl**. Check the Man page for each for detailed options and examples. **nfs4_getfacl** will list the access controls for a specified file or directory. **nfs4_setfacl** will create access controls for a directory or file, and **nfs4_editfacl** will let you change them. **nfs4_editfacl** simply invokes **nfs_setfacl** with the **-e** option. When editing access controls, you are placed in an editor where you can make your changes. For setting access controls you can read from a file, the standard input, or list the control entries on the command line.

The file and directory access controls are more refined that the standard permissions. The ACL entries follow the syntax described in detail on the **nfs4_acl** Man page. An ACL entry begins with an entry type such as an accept or deny entry (**A** or **D**); followed by an ACL flag, which can specify group or inheritance capability and then the principal to which the ACL is applied; and finally, the list of access options, such as **r** for read or **w** for write. The principal is usually a user URL that is to be permitted or denied access. You can also specify groups, but you need to set the **g** group flag. The special URLs OWNER@, GROUP@, and EVERYONE@ correspond to the owner, group, and other access used on standard permissions. The following example provides full access to the owner, but gives only read and execute access to the user **george@rabbit.com**. Group write and execute access is denied.

```
A::OWNER@:rwadtTnNcCy
A::george@rabbit.com:rxtncy
D:g:GROUP@:waxtc
```

In addition to read, write, and execute permissions (**r**,**w**,**x**), ACL lists also provide attribute reads (**t**,**n**) and attribute writes (**T**,**N**), as well as ACL read (**c**) and write (**C**) access. NFS read and write synchronization is enabled with the **y** option. The ability to delete files and directories is provided by the **d** option and for subdirectories with the **D** option. The **a** option lets you append data and create subdirectories. Keep in mind that **rtncy** are all read options, whereas **wadDTNC** are write options, while **x** remains the execute option. You will need **y** for any synchronized access. The **C** option in particular is very powerful as it allows the user to change the access controls (lowercase **c** allows only reading of the access controls).

Controlling Accessing to NFS Servers

You can use several methods to control access to your NFS server, such as using **hosts.allow** and **hosts.deny** to permit or deny access, as well as using your firewall to intercept access.

/etc/hosts.allow and /etc/hosts.deny

The **/etc/hosts.allow** and **/etc/hosts.deny** files are used to restrict access to services provided by your server to hosts on your network or on the Internet (if accessible). For example, you can use the **hosts.allow** file to permit access by certain hosts to your FTP server. Entries in the **hosts.deny** file explicitly deny access to certain hosts. For NFS, you can provide the same kind of security by controlling access to specific NFS daemons. The entries in the hosts.allow file are the same you specified in the **shares-admin** tool's Add Allow hosts window (Share Folder).

Portmapper Service

The first line of defense is to control access to the portmapper service. The portmapper tells hosts where the NFS services can be found on the system. Restricting access does not allow a remote host to even locate NFS. For a strong level of security, you should deny access to all hosts except those that are explicitly allowed. In the **hosts.deny** file, you place the following entry, denying access to all hosts by default. ALL is a special keyword denoting all hosts.

```
portmap:ALL
```

In the **hosts.allow** file, you then enter the hosts on your network, or any others that you want to permit access to your NFS server. Again, you specify the portmapper service and then list the IP addresses of the hosts you are permitting access. You can list specific IP addresses or a network range using a netmask. The following example allows access only by hosts in the local network, 192.168.0.0, and to the host 10.0.0.43. You can separate addresses with commas:

```
portmap: 192.168.0.0/255.255.255.0, 10.0.0.43
```

The portmapper is also used by other services such as NIS. If you close all access to the portmapper in **hosts.deny**, you will also need to allow access to NIS services in **hosts.allow**, if you are running them. These include ypbind and ypserver. In addition, you may have to add entries for remote commands like **ruptime** and **rusers**, if you are supporting them.

It is also advisable to add the same level of control for specific NFS services. In the **hosts.deny** file, you add entries for each service, as shown here:

```
mountd:ALL
rquotad:ALL
statd:ALL
lockd:ALL
```

Then, in the **hosts.allow** file, you can add entries for each service:

```
mountd:  192.168.0.0/255.255.255.0, 10.0.0.43
rquotad: 192.168.0.0/255.255.255.0, 10.0.0.43
statd:   192.168.0.0/255.255.255.0, 10.0.0.43
lockd:   192.168.0.0/255.255.255.0, 10.0.0.43
```

Netfilter Rules

You can further control access using Netfilter to check transmissions from certain hosts on the ports used by NFS services. portmapper uses port 111, and nfsd uses 2049. Netfilter is helpful if you have a private network that has an Internet connection and you want to protect it from the Internet. Usually a specific network device, such as an Ethernet card, is dedicated to the Internet connection. The following examples assume that device **eth1** is connected to the Internet. Any packets attempting access on port 111 or 2049 are refused.

```
iptables -A INPUT -i eth1 -p 111 -j DENY
iptables -A INPUT -i eth1 -p 2049 -j DENY
```

To enable NFS for your local network, you will have to allow packet fragments. Assuming that **eth0** is the device used for the local network, you could use the following example:

```
iptables -A INPUT -i eth0 -f -j ACCEPT
```

Mounting NFS File Systems: NFS Clients

Once NFS makes directories available to different hosts, those hosts can then mount those directories on their own systems and access them. The host needs to be able to operate as an NFS client. Current Linux kernels all have NFS client capability built in. This means that any NFS client can mount a remote NFS directory that it has access to by performing a simple mount operation.

Mounting NFS Automatically: /etc/fstab

You can mount an NFS directory either by an entry in the **/etc/fstab** file or by an explicit **mount** command. You have your NFS file systems mounted automatically by placing entries for them in the **/etc/fstab** file. An NFS entry in the **/etc/fstab** file has a mount type of NFS. An NFS file system name consists of the hostname of the computer it is located on, followed by the pathname of the directory where it is mounted. The two are separated by a colon. For example, **rabbit.trek.com:/home/project** specifies a file system mounted at **/home/project** on the **rabbit.trek.com** computer. The format for an NFS entry in the **/etc/fstab** file follows. The file type for NFS versions 1 through 3 is **nfs**, whereas for NFS version 4 it is **nfs4**.

```
host:remote-directory    local-directory      nfs   options   0   0
```

Option	Description
rsize=*n*	The number of bytes NFS uses when reading files from an NFS server. The default is 1,024 bytes. A size of 8,192 can greatly improve performance.
wsize=*n*	The number of bytes NFS uses when writing files to an NFS server. The default is 1,024 bytes. A size of 8,192 can greatly improve performance.
timeo=*n*	The value in tenths of a second before sending the first retransmission after a timeout. The default value is seven-tenths of a second.
retry=*n*	The number of minutes to retry an NFS mount operation before giving up. The default is 10,000 minutes (one week).
retrans=*n*	The number of retransmissions or minor timeouts for an NFS mount operation before a major timeout (default is 3). At that time, the connection is canceled or a “server not responding” message is displayed.
soft	Mount system using soft mount.
hard	Mount system using hard mount. This is the default.
intr	Allows NFS to interrupt the file operation and return to the calling program. The default is not to allow file operations to be interrupted.
bg	If the first mount attempt times out, continues trying the mount in the background. The default is to fail without backgrounding.
tcp	Mounts the NFS file system using the TCP protocol, instead of the default UDP protocol.

Table 25-2: NFS Mount Options

You can also include several NFS-specific mount options with your NFS entry. You can specify the size of datagrams sent back and forth and the amount of time your computer waits for a response from the host system. You can also specify whether a file system is to be hard-mounted or soft-mounted. For a *hard-mounted* file system, your computer continually tries to make contact if for some reason the remote system fails to respond. A *soft-mounted* file system, after a specified interval, gives up trying to make contact and issues an error message. A hard mount is the default. A system making a hard-mount attempt that continues to fail will stop responding to user input as it tries continually to achieve the mount. For this reason, soft mounts may be preferable, as they will simply stop attempting a mount that continually fails. Table 25-2 and the Man pages for **mount** contain a listing of these NFS client options. They differ from the NFS server options indicated previously.

An example of an NFS entry follows. The remote system is **rabbit.mytrek.com**, and the file system is mounted on **/home/projects**. This file system is to be mounted on the local system as the **/home/dylan/projects** directory. The **/home/dylan/projects** directory must already be created on the local system. The type of system is NFS, and the **timeo** option specifies the local system waits up to 20 tenths of a second (two seconds) for a response. The mount is a soft mount and can be interrupted by NFS.

```
rabbit.mytrek.com:/home/projects /home/dylan/projects  nfs  soft,intr,timeo=20
```

Mounting NFS Manually: mount

You can also use the **mount** command with the **-t nfs** option to mount an NFS file system explicitly. For a NFSv4 file system you use **-t nfs4**. To mount the previous entry explicitly, use the following command:

```
mount -t nfs -o soft,intr,timeo=20 rabbit.mytrek.com:/home/projects \
/home/dylan/projects
```

You can, of course, unmount an NFS directory with the **umount** command. You can specify either the local mountpoint or the remote host and directory, as shown here:

```
umount /home/dylan/projects
umount  rabbit.mytrek.com:/home/projects
```

Mounting NFS on Demand: autofs

You can also mount NFS file systems using the automount service, autofs. This requires added configuration on the client's part. The autofs service will mount a file system only when you try to access it. A directory change operation (**cd**) to a specified directory will trigger the mount operation, mounting the remote file system at that time.

The autofs service is configured using a master file to list map files, which in turn lists the file systems to be mounted. The **/etc/auto.master** file is the autofs master file. The master file will list the root pathnames where file systems can be mounted along with a map file for each of those pathnames. The map file will then list a key (subdirectory), mount options, and the file systems that can be mounted in that root pathname directory. On some distributions, the **/auto** directory is already implemented as the root pathname for file systems automatically mounted. You can add your own file systems in the **/etc/auto.master** file along with your own map files, if you wish. You

will find that the **/etc/auto.master** file contains the following entry for the **/auto** directory, listing **auto.misc** as its map file:

```
/auto    auto.misc   --timeout 60
```

Following the map file, you can add options, as shown in the preceding example. The **timeout** option specifies the number of seconds of inactivity to wait before trying to automatically unmount.

In the map file, you list the key, the mount options, and the file system to be mounted. The key will be the subdirectory on the local system where the file system is mounted. For example, to mount the **/home/projects** directory on the **rabbit.mytrek.com** host to the **/auto/projects** directory, you use the following entry:

```
projects  soft,intr,timeo=20   rabbit.mytrek.com:/home/projects
```

You can also create a new entry in the master file for an NFS file system, as shown here:

```
/myprojects    auto.myprojects   --timeout 60
```

You then create an **/etc/auto.myprojects** file and place entries in it for NFS files system mounts, like the following:

```
dylan    soft,intr,rw   rabbit.mytrek.com:/home/projects
newgame  soft,intr,ro   lizard.mytrek.com:/home/supergame
```

Network Information Service: NIS

On networks supporting NFS, many resources and devices are shared by the same systems. Normally, each system needs its own configuration files for each device or resource. Changes entail updating each system individually. However, NFS provides a special service called the Network Information System (NIS) that maintains such configuration files for the entire network. For changes, you need only to update the NIS files. NIS works for information required for most administrative tasks, such as those relating to users, network access, or devices. For example, you can maintain user and password information with an NIS service, having only to update those NIS password files.

Note: NIS+ is a more advanced form of NIS that provides support for encryption and authentication. However, it is more difficult to administer.

NIS was developed by Sun Microsystems and was originally known as Sun's Yellow Pages (YP). NIS files are kept on an NIS server (NIS servers are still sometimes referred to as YP servers). Individual systems on a network use NIS clients to make requests from the NIS server. The NIS server maintains its information on special database files called *maps*. Linux versions exist for both NIS clients and servers. Linux NIS clients easily connect to any network using NIS.

Note: Instead of NIS, many networks now use LDAP to manage user information and authentication.

The NIS client is installed as part of the initial installation on most Linux distributions. NIS client programs are ypbind (the NIS client daemon), ypwhich, ypcat, yppoll, ypmatch, yppasswd, and ypset. Each has its own Man page with details of its use. The NIS server programs are ypserv (the NIS server), ypinit, yppasswdd, yppush, ypxfr, and netgroup—each also with its own Man page.

You can start and stop the **ypbind** client daemon and the **ypserv** NIS server with the **service** command. Alternatively, you can use system-config-services to start and stop the NIS client and server daemons.

```
service ypbind start
service ypserv start
```

Note: You can use system-config-authentication to specify the remote NIS server on your network.

NIS Servers

You have significant flexibility when setting up NIS servers. If you have a small network, you may need only one NIS domain, for which you would have one NIS server. For larger networks, you can divide your network into several NIS domains, each with its own server. Even if you only have one domain, you may want several NIS slave servers. For an NIS domain, you can have a master NIS server and several NIS slave servers. The slave servers can act as backups, in case the master server goes down. A slave server only contains copies of the configuration files set up on the NIS master server.

Configuring an NIS server involves several steps, listed here:

1. Define the NIS domain name that the NIS server will work for.
2. Start the **ypserv** daemon.
3. In the **/var/yp/Makefile** file, set any NIS server options and specify the configuration files to manage.
4. Use **/usr/lib/ypinit** to create the NIS versions of the configuration files.

Defining NIS Domain

You first have to define an NIS domain name. You can have the NIS domain defined whenever you start up your system, by defining the NIS_DOMAIN variable in the **/etc/sysconfig/network** file. To this variable, you assign the name you want to give your NIS domain. The following example defines the NIS domain called **myturtles.nis**:

```
NIS_DOMAIN=myturtles.nis
```

When first setting up the server, you may want to define your NIS domain name without having to restart your system. You can do so with the **domainname** command, as shown here:

```
domainname myturtles.nis
```

NIS server options are kept in the **/etc/ypserv.conf** file. Check the man page for that file for details. You can start the NIS server with the **ypserv** startup script:

```
service ypserv start
```

Setting NIS Server Options

Next edit the **/var/yp/Makefile** file to select the configuration files that the NIS server will maintain, along with setting any NIS server options. Standard options as well as most commonly used configuration files are usually already set up.

NIS server options are listed first. The **NOPUSH** option will be set to true, indicating that there are no slave NIS servers. If you are setting up any slave NIS servers for this domain, you will have to set this option to false:

```
NOPUSH = true
```

The minimum user and group IDs are set to 500. These are set using the **MINUID** and **MINGID** variables:

```
MINUID=500
MINGID=500
```

Most distributions use a shadow password and shadow group files to encrypt passwords and groups; the **MERGE_PASSWD** and **MERGE_GROUP** settings will be set to true. NIS will merge shadow password information into its password file:

```
MERGE_PASSWD=true
MERGE_GROUP=true
```

The directories where NIS will find password and other configuration files are then defined using the **YPSRCDIR** and **YPPWDIR** variables. Normally, the **/etc** directory holds your configuration files:

```
YPSRCDIR = /etc
YPPWDDIR = /etc
```

Then the configuration files that NIS can manage are listed. Here, you will find entries like **PASSWD** for password, **GROUP** for your groups, and **PRINTCAP** for your printers. A sample of the entries are shown here:

```
GROUP       = $(YPPWDDIR)/group
PASSWD      = $(YPPWDDIR)/passwd
SHADOW      = $(YPPWDDIR)/shadow
GSHADOW     = $(YPPWDDIR)/gshadow
ALIASES     = /etc/aliases
ETHERS      = $(YPSRCDIR)/ethers     # ethernet addresses (for rarpd)
BOOTPARAMS  = $(YPSRCDIR)/bootparams # for booting Sun boxes (bootparamd)
HOSTS       = $(YPSRCDIR)/hosts
NETWORKS    = $(YPSRCDIR)/networks
PRINTCAP    = $(YPSRCDIR)/printcap
PROTOCOLS   = $(YPSRCDIR)/protocols
```

Specifying Shared Files

The actual files that are shared on the network are listed in the **all:** entry, which follows the list of configuration files. Only some of the files defined are listed as shared, those listed in the first line after **all:**. The remaining lines are automatically commented out (with a preceding **#** sign). You can add files by removing the **#** sign or moving their entries to the first line.

```
all:  passwd group hosts rpc services netid protocols mail \
      # netgrp shadow publickey networks ethers bootparams printcap \
      # amd.home auto.master auto.home auto.local passwd.adjunct \
      # timezone locale netmasks
```

Be sure not to touch the remainder of the Makefile.

Creating the NIS Database

You then enter the **ypinit** command with the **-m** option to create the NIS database consisting of the NIS configuration files. Your NIS server will be detected, and then you will be asked to enter the names of any slave NIS servers used on this NIS domain. If there are any, enter them. When you are finished, press CTRL-D. The NIS database files are then created.

```
ypinit -m
```

For an NIS slave server, you would use

```
ypinit -s masterhost
```

Should you receive the following error, it most likely means that your NIS server was not running. Be sure to start ypserv before you run **ypinit**.

```
failed to send 'clear' to local ypserv: RPC: Program not registeredUpdating
```

If you later need to update your NIS server files, you would change to the **/var/yp** directory and issue the **make** command.

```
cd /var/yp
make
```

Controlling Access

The **/var/yp/securenets** file enables access by hosts to your NIS server. Hosts can be referenced by network or individually. Entries consist of a subnet mask and an IP address. For example, you could give access to all the hosts in an local network with the following entry:

```
255.255.255.0  192.168.1.0
```

For individual hosts, you can use the mask 255.255.255.255 or just the term "host," as shown here:

```
host   192.168.1.4
```

Controlling how different hosts access NIS shared data is determined in **/etc/ypserv.conf**.

Netgroups

You can use NIS to set up netgroups, which allow you to create network-level groups of users. Whereas normal groups are created locally on separate hosts, an NIS netgroup can be used for network-wide services. For example, you can use NIS netgroups to control access to NFS file systems. Netgroups are defined in the **/etc/netgroup** file. Entries consist of a netgroup name followed by member identifiers consisting of three segments: the host, the user, and the NIS domain:

```
group    (host, user, NIS-domain) (host, user, NIS-domain) ...
```

For example, in the NIS domain **myturtles.nis**, to define a group called **myprojects** that consist of the user **chris** on the host **rabbit**, and the user **george** on the host **lizard.mytrek.com**, you would use the following:

```
myprojects (rabbit, chris, myturtles.nis) \
                          (lizard.mytrek.com, george, myturtles.nis)
```

A blank segment will match on any value. The following entry includes all users on the host **rabbit**:

```
newgame (rabbit,,myturtles.ni)
```

If your use of a group doesn't need either a user or a host segment, you can eliminate one or the other using a hyphen (-). The following example generates a netgroup consisting just of hostnames, with no usernames:

```
myservers (rabbit,-,) (turtle.mytrek.com,-,)
```

You can then reference different netgroups in various configuration files by prefixing the netgroup name with an `@` sign, as shown here:

```
@newgame
```

NIS Clients

For a host to use NIS on your network, you first need to specify your NIS domain name on that host. In addition, your NIS clients need to know the name of your NIS server. If you installed Linux on a network already running NIS, you may have already entered this information during the installation process.

Specifying the NIS Domain and Server

You can specify your NIS domain name and server with the authconfg-gtk tool, which you can access from the System Settings window. In that window, select Authentication. This opens the Authentication Configuration window. On the User Information panel, click the Configure NIS button to open a dialog where you can enter the name of the NIS domain as well as the NIS server. Be sure to also enable NIS on the User Information panel. The NIS domain will be saved in the **/etc/sysconfig/network** file, and the NIS server, in the **/etc/yp.conf** file.

Accessing the Server

Each NIS client host on your network then has to run the ypbind NIS client to access the server. In the client's **/etc/yp.conf** file, you need to specify the NIS server it will use. The following entry would reference the NIS server at 192.168.1.1:

```
ypserver 192.168.1.1
```

Alternatively, you can specify the NIS domain name and the server it uses:

```
domain mydomain.nis  server servername
```

The authconfg-gtk tool will make the following entry in **/etc/yp.conf** for the **myturtle.nis** NIS domain using the **turtle.mytrek.com** server:

```
domain myturtles.nis server turtle.mytrek.com
```

To start the NIS client, you run the **ypbind** script:

```
service ypbind start
```

Then, to check that all is working, you can use **ypcat** to try to list the NIS password file:

```
ypcat passwd.
```

You can use **ypcat** to list any of the NIS configuration files. The **ypwhich** command will display the name of the NIS server your client is using. **ypmatch** can be used to find a particular entry in a configuration file.

```
ypmatch cecelia passwd.
```

Users can change their passwords in the NIS **passwd** file by using the **yppasswd** command. It works the same as the **passwd** command. You will also have to have the **yppasswdd** daemon running.

Specifying Configuration Files with nsswitch.conf

To ensure that the client accesses the NIS server for a particular configuration file, you should specify **nisplus** in file's entry in the **/etc/nsswitch.conf** file. The **nisplus** option refers to the NIS version 3. The **nis** option refers to the older NIS version 2. The **/etc/nsswitch.conf** file specifies where a host should look for certain kinds of information. For example, the following entry says to check the local configuration files (**files**) first and then the NIS server (**nisplus**) for password data:

```
passwd:   files nisplus
```

The **files** designation says to first use the system's own files, those on the local host. **nis** says to look up entries in the NIS files, accessing the NIS server. **nisplus** says to use NIS+ files maintained by the NIS+ server. **dns** says to perform DNS lookups; it can only be used on files like **hosts** that contain hostnames. These are some standard entries:

```
passwd:     files nisplus
shadow:     files nisplus
group:      files nisplus

hosts:      files nisplus dns
bootparams: nisplus [NOTFOUND=return] files

ethers:     files
netmasks:   files
networks:   files
protocols:  files nisplus
rpc:        files
services:   nisplus
netgroup:   nisplus
publickey:  nisplus
automount:  files nisplus
aliases:    files nisplus
```

26. Samba

Most local and home networks may include some systems working on Microsoft Windows and others on Linux. You may need to let a Windows computer access a Linux system or vice versa. Windows, due to its massive market presence, tends to benefit from both drivers and applications support not found for Linux. Though there are equivalent applications on Linux, many of which are as good or better, some applications run best on Windows, if for no other reason than that the vendor only develops drivers for Windows.

One solution is to use the superior server and storage capabilities of Linux to manage and hold data, while using Windows systems with their unique applications and drivers to run applications. For example, you could use a Linux system to hold pictures and videos, while using Windows systems to show or run them. Video or pictures could be streamed through your router to the system that wants to run them. In fact many commercial DVR systems use a version of Linux to manage video recording and storage. Another use would be to enable Windows systems to use devices like printers that may be connected to a Linux system, or vice versa.

With Samba, you can connect your Windows clients on a Microsoft Windows network to services such as shared files, systems, and printers controlled by the Linux Samba server and, at the same time, allow Linux systems to access shared files and printers on Windows systems. Samba is a collection of Linux tools that allow you to communicate with Windows systems over a Windows network. In effect, Samba allows a Linux system or network to act as if it were a Windows server, using the same protocols as used in a Windows network. Whereas most Unix and Linux systems use the TCP/IP protocol for networking, Microsoft networking with Windows uses a different protocol, called the Server Message Block (SMB) protocol that implements a local area network (LAN) of PCs running Windows. SMB makes use of a network interface called Network Basic Input Output System (NetBIOS) that allows Windows PCs to share resources, such as printers and disk space. One Windows PC on such a network can access part of another Windows PC's disk drive as if it were its own. SMB was originally designed for small LANs. To connect it to larger networks, including those with Unix systems, Microsoft developed the Common Internet File System (CIFS). CIFS still uses SMB and NetBIOS for Windows networking. Wanting to connect his Linux system to a Windows PC, Andrew Tridgell wrote a SMB client and server that he called Samba. Samba allows Unix and Linux systems to connect to such a Windows network; as if they were Windows PCs. Unix systems can share resources on Windows systems as if they were just another Windows PC. Windows PCs can also access resources on Unix systems as if they were Windows systems. Samba, in effect, has become a professional-level, open source, and free version of CIFS. It also runs much faster than CIFS. Samba effectively enables you to use a Linux or Unix server as a network server for a group of Windows machines operating on a Windows network. You can also use it to share files on your Linux system with other Windows PCs, or to access files on a Windows PC from your Linux system, as well as between Windows PCs. On Linux systems, a **cifs** file system enables you, in effect, to mount a remote SMB-shared directory on your own file system. You can then access it as if it were a directory on your local system.

You can obtain extensive documentation and current releases from the Samba Web and FTP sites at **www.samba.org** and **ftp.samba.org**. Samba HOW-TO documentation is also available at **www.tldp.org**. Other information can be obtained from the SMB newsgroup, **comp.protocols.smb**. Packages can be obtained from your distribution software repositories.

Extensive documentation is provided with the software package and installed on your system in the **/usr/share/doc/samba-doc** directory. Be sure to install the **samba-doc** package. The **htmldocs** subdirectory holds various documentation. All are in Web page format. Documentation

includes the HOWTO, By Example, Using Samba, and Developers Guide. The examples include sample **smb.conf** files for different kinds of configuration. For PDF versions install the **samba-doc-pdf** package, **/usr/share/doc/samba-doc-pdf**.

On Fedora, Samba software is incorporated into several packages, with configuration tools such as SWAT and system-config-samba (see Table 26-1). By selecting the samba server package, needed supporting packages like smbclient and samba-common will be automatically selected. Documentation and configuration tools have to be selected manually.

Package name	Description
samba	The Samba server
samba-common	Samba Fedora configuration files and support tools
samba-doc	Documentation for Samba, including examples
system-config-samba	Samba GUI configuration tool from Red Hat
samba-doc-pdf	PDF versions for Samba documentation
swat	SWAT Samba Web interface for Samba configuration
smbclient	Samba clients for accessing Windows shares.
smbfs	Mount and unmount tools for Samba shares
kdenetwork-filesharing	Implements Samba file sharing by KDE
shares-admin	GNOME Samba file sharing support, installed with GNOME Desktop.

Table 26-1: Samba packages on Fedora

Samba Applications

The Samba software package consists of two server daemons and several utility programs (see Table 26-2). One daemon, **smbd**, provides file and printer services to SMB clients and other systems, such as Windows, that support SMB. The **nmbd** is a daemon that provides NetBIOS name resolution and service browser support. The **smbclient** tool provides FTP-like access by Linux clients to Samba services. **mount.cifs** and **umount.cifs** enable Linux clients to mount and unmount Samba shared directories (used by the `mount` command with the `-t samba` option). The smbstatus utility displays the current status of the SMB server and who is using it. You use **testparm** to test your Samba configuration. `smbtar` is a shell script that backs up SMB/CIFS-shared resources directly to a Unix tape drive. You use **nmblookup** to map the NetBIOS name of a Windows PC to its IP address. The primary Samba configuration tool for Fedora is system-config-samba, which enables you to use a GUI interface to create and maintain your Samba configuration file, **/etc/samba/smb.conf**. Alternatively, the Samba Web administration tool (SWAT) Samba to configure Samba. Configuration files are kept in the **/etc/samba** directory.

Samba provides four main services: file and printer services, authentication and authorization, name resolution, and service announcement. The SMB daemon, **smbd**, provides the file and printer services, as well as authentication and authorization for those services. This means users on the network can share files and printers. You can control access to these services by

requiring users to provide a password. When users try to access a shared directory, they are prompted for a password. Control can be implemented in share mode or user mode. The *share* mode sets up one password for the shared resource and then enables any user who has that password to access it. The *user* mode provides a different password for each user. Samba maintains its own password file for this purpose.

Application	Description
system-config-samba	Fedora Samba administration tool for configuring **smb.conf** with a GNOME GUI interface
SWAT	Samba Web administration tool for configuring **smb.conf** with a Web browser
smbd	Samba server daemon that provides file and printer services to SMB clients
nmbd	Samba daemon that provides NetBIOS name resolution and service browser support
winbindd	Uses authentication services provided by Windows domain
smbclient	Provides FTP-like access by Linux clients to Samba services
mount.cifs	Mounts Samba share directories on Linux clients (used by the `mount` command with the `-t cifs` option)
smbpasswd	Changes SMB-encrypted passwords on Samba servers
smbstatus	Displays the current status of the SMB network connections
smbrun	Interface program between **smbd** and external programs
testparm	Tests the Samba configuration file, **smb.conf**
smbtar	Backs up SMB/CIFS-shared resources directly to a Unix tape drive
nmblookup	Maps the NetBIOS name of a Windows PC to its IP address
/etc/init.d/samba	Samba init script to start, stop, and restart the Samba server.

Table 26-2: Samba Applications

Name resolution and service announcements are handled by the nmbd server. Name resolution essentially resolves NetBIOS names with IP addresses. Service announcements, also known as *browsing,* are the way a list of services available on the network is made known to the connected Windows PCs (and Linux PCs connected through Samba).

Samba also includes the **winbindd** daemon, which allows Samba servers to use authentication services provided by a Windows domain. Instead of a Samba server maintaining its own set of users to allow access, it can make use of a Windows domain authentication service to authenticate users.

Setting Up Samba

To allow Windows to access a Linux system, as well as Linux to access a Windows system, you use the Samba server. First be sure that Samba is installed along with the Services tool. Open the Add/Remove Software tool in the Administration menu (PackageKit). Click the Server

entry or search for samba. On the left pane, and scroll down and locate the **Samba** and **system-config-samba** packages. If the item is not checked, check it, then click Install to install it.

Note: It is possible to set up Samba shared directories with the GNOME Shared Folders tool. However this tool does not provide for user level security which is now deprecated. It only provides very open share level access to any user.

Samba has two methods of authentication, shares and users. User authentication requires that there be corresponding accounts in the Windows and Linux systems. They can have the same name, though a Windows user can be mapped to a Linux account. A share can be made open to any user and function as an extension of the user's storage space.

Use the system-config-services configuration (System | Administration | Services) to have Samba start up automatically (see Figure 26-1). Select the **smb** and **nmb** entries, and click the Enable button for each.

To set up simple file sharing on a Linux system, you first need to configure your Samba server. You can do this by directly editing the **/etc/samba/samba.conf** file or by using the **system-config-samba** configuration tool (System | Administration | Samba). If you just edit the **/etc/samba/samba.conf** file, you first need to specify the name of your Windows network.

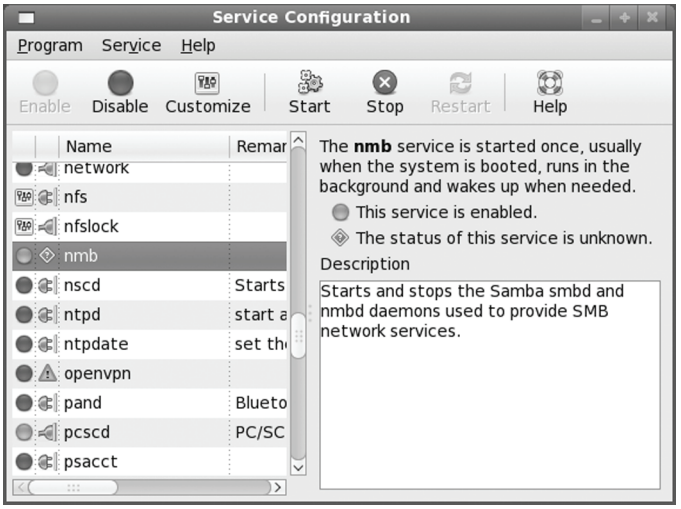


Figure 26-1: system-config-services for samba, System | Administration | Services

Starting Samba

Once installed, Samba is normally configured to start up automatically. You can turn this option on or off using the system-config-services (System | Administration | Services). To manually start Samba, select the **smb** and then the **nmb** entries, and click the Start button for each.

For a simple Samba setup, you can use Fedora system-config-samba or SWAT to configure your **/etc/samba/smb.conf** file. If you make changes, you must restart the Samba server to have the changes take effect. To restart Samba with your new configuration, use the **smb** and **nmb** init script with the **restart** option, **/etc/init.d/smb** and **/etc/init.d/nmb**. The start, stop, and

restart options will start, stop, and restart the server. Run the following commands from a terminal window to restart Samba.

```
service nmb restart
service smb restart
```

Tip: The Samba server needs to run both the **nmbd** and the **smbd** servers. Without the **nmbd** server, Windows cannot detect your Samba server. They are started by **/etc/init.d/smb** and **/etc/init.d/nmb** scripts.

Firewalls

The IPtables firewall prevents browsing Samba and Windows shares from your Linux desktop. To work around this restriction, you use system-config-firewall level (System | Administration | Firewall). Select the Samba entry in the Trusted services pane. The firewall on your Windows computer and on your network router could, in turn, block Samba server access. Be your Windows and router firewalls are configured to accept local IP address assigned by the router for computers on your network.

system-config-samba

The system-config-samba tool provides a simple means to configure your Samba server, adding Samba users and specifying Samba shares . It will also automatically configure all the printers on your Linux system as Samba shared printers, allowing you to use them from any connected Windows system. You can start system-config-samba from the Samba entry in the System | Administration menu (see Figure 26-2). The system-config-samba tool will list all the shares for your server. You can use buttons at the top to manage your shares, adding new ones or deleting current ones. If you delete, the actual directories are not removed; they just lose their status as shared directories.

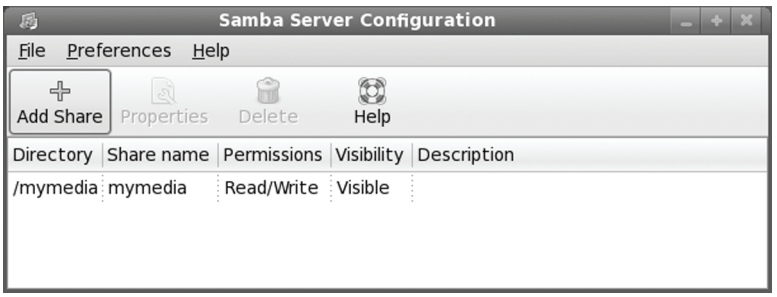


Figure 26-2: system-config-samba

Note: As an alternative to system-config-samba, you can use SWAT, a Samba Web browser based tool that supports complex configurations. Install the swat package and the xinetd package. Be sure to also enable swat with the chkconfig tool. SWAT will be accessible at port 901 on the system hosting the samba server.

Samba server configuration

To configure your Samba server, select Server Settings from the system-config-samba Preferences menu. This opens a window with two panels; Basic and Security (see Figure 26-3). On

the Basic panel, you enter the Samba server workgroup name. This will be the same name used as the workgroup by all your Windows systems. The default names given by Windows are MSHOME or WORKGROUP. Use the name already given to your Windows network. For home networks, you can decide on your own. Just make sure all your computers use the same network name. Check your Windows Control Panel's System applet to make sure.

The description is the name you want displayed for your Samba server on your Windows systems. On the Security panel, you specify the authentication mode, the password encryption option, and the name of the guest account, along with the authentication server.

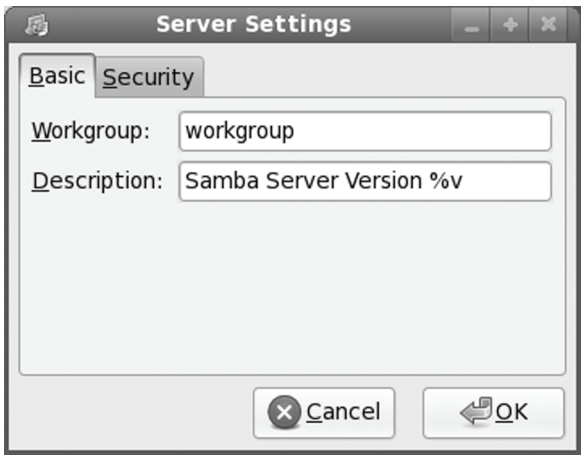


Figure 26-3: Samba Server Settings

By default User security is used. You could also use share or server security. these are more open, but both have been deprecated and may be dropped in later versions.

- The authentication mode specifies the access level, which can be user, share, server, ADS, or domain. User-level access restricts access by user password, whereas share access opens access to any guest.
- Normally, you would elect to encrypt passwords, rather than have them passed over your network in plain text.
- The Guest user is the name of the account used to allow access to shares or printers that you want open to any user, without having to provide a password. The pop-up menu will list all your current users, with "No Guest Account" as the selected default. Unless you want to provide access by everyone to a share, you would not have a Guest account.

Adding Samba Users

For user authentication you will have to associate a Windows user with a particular Samba user. Samba maintains its own password listing for users. To provide a user Samba access, you need to register the user as a Samba user. Select Samba Users from the Preferences menu to open the Samba Users window, clicking the Add User button. Here you enter the Unix Username, the Windows Username, and the Samba Password. There is an additional box for confirming the

Samba password. The Unix Username is a pop-up window listing all the users on your Samba server.

Select Samba Users in the system-config-samba Preferences menu. This opens the Samba Users window listing all current Samba users. These correspond to user accounts already set up on your system. To add a Samba user, click the Add User button on the right. This opens a Create Samba User window with four entries (see Figure 26-4). Here you enter the Unix Username, the Windows Username, and the Samba Password. There is an additional box for confirming the Samba password. The Unix Username is a pop-up window listing all the users on the Linux system hosting your Samba server.

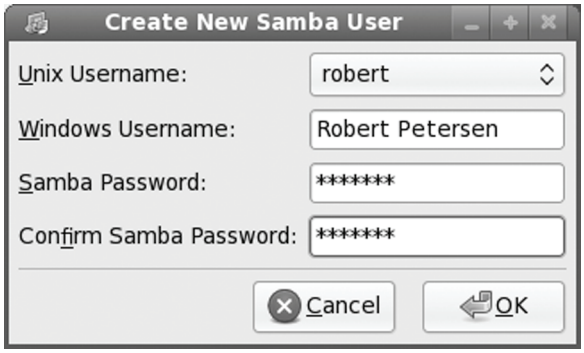


Figure 26-4: Adding Samba users

Select a Linux user to us from the Unix Username pop up menu, then enter the corresponding Windows user. The Windows username can be different from the Linux account (Linux user name). You then a password that Windows user can use to access Linux. This is the Samba password for that user. Samba maintains its own set of passwords that users will need to access a Samba share. When a Windows user wants to access a Samba share, they will need their Samba password.

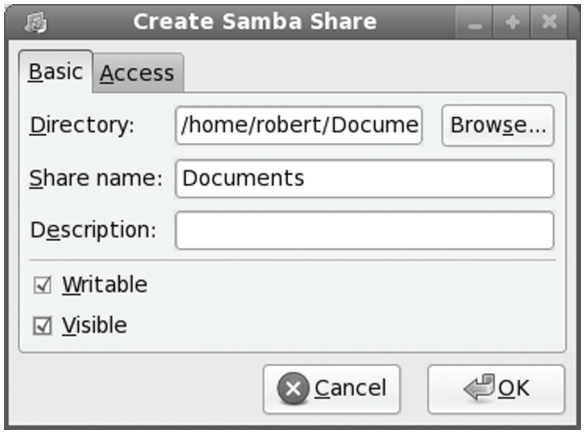


Figure 26-5: Create Samba Shares

Once you create a Samba user, its name will appear in the list of Samba users on the Samba Users window. To later modify or delete a Samba user, Use the same Samba Users window, select the user from the list, and click the Edit User button to change entries like the password, or click the Delete User button to remove the Samba user.

Adding Samba shares on Linux systems

To set up a simple share, click Add Share, which opens a Create A Share window. On the Basic panel you select the Linux directory to share (click Browse to find it), and then specify whether it will be writable and visible. You also provide a name for the share, as well as any description you want (see Figure 26-5).

Figure 26-6: Create Samba Shares: Access

On the Access panel, you can restrict access to certain users or allow access to all users. All Samba users on your system will be listed with check boxes where you can select those you want to give access (see Figure 26-6).

Once created, the share will appear in the main system-config-samba window. The share's directory, share name, its visibility, read/write permissions, and description will be shown. To later modify a share, click on its entry and then click on the Properties menu. This opens an Edit samba share window with the same Basic and Access panels you used to create the share.

Accessing Samba shares

When a Windows user wants to access the share on the Linux system, they open their My Network Places and then "Add a network place" to add a network place entry for the share, or View workgroup computers to see computers on your Windows network. Selecting the Linux Samba server will display your Samba shares. To access the share, the user will be required to enter in the user name and the Samba password. You have the option of having the username and password remembered for automatic access.

On your Linux system, to restart Samba with your new configuration use the Services tool, restarting both **nmb** and **smb** (System | Administration | Services). Be sure that the firewall on your Windows system is not blocking Samba. Run system-config-firewall (System | Administration |

Firewall) and choose the Trusted services panel. Make sure that the Samba service is checked, allowing Samba to operate on your system.

SWAT

SWAT is a network-based Samba configuration tool that uses a Web page interface to enable you to configure your **smb.conf** file. Be sure you have installed the **swat** package from the Fedora repository. SWAT is an easy way to configure your Samba server, providing the full range of configuration options. SWAT provides a simple-to-use Web page interface with buttons, menus, and text boxes for entering values. A simple button bar across the top enables you to select the sections you want to configure. A button bar is even there to add passwords. To see the contents of the **smb.conf** file as SWAT changes it, click View. The initial screen (HOME) displays the index for Samba documentation. One of SWAT's more helpful features is its context-sensitive help. For each parameter and option SWAT displays, you can click a Help button to display a detailed explanation of the option and examples of its use.

Activating SWAT

SWAT is normally installed with Samba. SWAT is an xinetd service. As an xinetd service, it will be listed in the **/etc/services** and **/etc/xinetd.d/swat** files. The SWAT program uses port 901, as designated in the **/etc/services** file and shown here:

```
swat 901/tcp # Samba Web Administration Tool
```

As an xinetd service, SWAT will have its own xinetd file in the **/etc/xinetd.d** directory, **/etc/xinetd.d/swat**. SWAT is turned off by default, and its **disable** option is set to **yes**. To use SWAT, you will have to change the **disable** option to **no** as shown here:

```
# default: off
# description: SWAT is the Samba Web Admin Tool. Use swat \
#      to configure your Samba server. To use SWAT, \
#      connect to port 901 with your favorite web browser.
service swat
{
      disable = no
      port = 901
      socket_type = stream
      wait = no
      only_from = 127.0.0.1
      user = root
      server = /usr/sbin/swat
      log_on_failure += USERID
}
```

You can do this by using either **chkconfig** or the Service Configuration tool to turn on the SWAT service or by manually editing the **/etc/xinetd.d/swat** file and changing the **disable** option to **no**. **chkconfig** will edit the **/etc/xinetd.d/swat** file for you, making this change (see Chapter 3 for more information about **chkconfig**). The following example shows how you would enable SWAT with the **chkconfig** command:

```
chkconfig swat on
```

With **chkconfig**, you will not have manually restart the xinetd server. However, if you manually edit the file, you will also have to restart the server to have the change take effect. You can do this simply using the **xinetd** script, as shown here:

```
service xinetd restart
```

Before you use SWAT, back up your current **smb.conf** file. SWAT overwrites the original, replacing it with a shorter and more concise version of its own. The **smb.conf** file originally installed lists an extensive number of options with detailed explanations. This is a good learning tool, with excellent examples for creating various kinds of printer and directory sections. Simply make a backup copy:

```
cp /etc/samba/smb.conf /etc/samba/smb.bk
```

Accessing SWAT

You can start SWAT by opening your browser and enter the IP address 127.0.0.1 with port 901 to access SWAT.

```
http://127.0.0.1:901
```

Instead of 127.0.0.1 you can use **localhost.localdomain**.

```
http://localhost.localdomain:901
```

You can start SWAT from a remote locate by entering the address of the Samba server it is running on, along with its port (901), into a Web browser.

You are first asked to enter a username and a password. To configure Samba, you need to enter **root** and the root password.

Page	Description
HOME	SWAT home page listing documentation resources.
GLOBALS	Configures the global section for Samba.
SHARES	Selects and configures directories to be shared (shares).
PRINTERS	Sets up access to printers.
WIZARD	Quick server setup, rewrites original smb.conf file removing all comments and default values.
STATUS	Checks the status of the Samba server, both smbd and nmbd; lists clients currently active and the actions they are performing. You can restart, stop, or start the Samba server from this page.
VIEW	Displays the **smb.conf** configuration file.
PASSWORD	Sets up password access for the server and users that have access.

Table 26-3: SWAT Configuration Pages

SWAT Configuration Pages

The main SWAT page is displayed with a button bar, with buttons for links for HOME, GLOBAL, SHARES, PRINTERS, STATUS, VIEW, and PASSWORD (see Table 26-3). You can use STATUS to list your active SMB network connections.

For the various sections, SWAT can display either a basic or advanced version. The basic version shows only those entries needed for a simple configuration, whereas the advanced version shows all the possible entries for that type of section. Buttons labeled Advanced and Basic appear at the top of the section page for toggling between the advanced or basic versions . Section pages for printers and shares have added buttons and a menu for selecting the particular printer or share you want to configure. The term "share," as its used here, refers to directories you want to make available through Samba. When you click the SHARES button, you initially see only a few buttons displayed at the top of the SHARES page. You use these buttons to create new sections or to edit sections already set up for shares. To set up a new Share section, you enter its name in the box next to the Create Share button and then click that button. The new share name appears in the drop-down menu next to the Choose Share button. Initially, this menu is blank. Click its drop-down symbol to display the list of current Share sections. Select the one you want, and then click the Choose Share button. The page then displays the entries for configuring a share. For a new share, these are either blank or default values. For example, to select the Homes section that configures the default setting for user home directories, click the drop-down menu, where you find a Homes entry. Select it, and then click the Choose Share button. The entries for the Homes section are displayed. The same process works for the Printers page, where you can select either the Printers section or the Create sections for particular printers.

Note: For Samba to use a printer, it first has to be configured on your system as either a local or network printer. Keep in mind that a network printer could be a printer connected to a Windows system.

There is a Help link next to each entry. Such a link displays a Web page showing the Samba documentation for **smb.conf**, positioned at the appropriate entry.

When you finish working on a section, click the Commit Changes button on its page to save your changes. Do this for each separate page you work on, including the GLOBALS page. Clicking Commit Changes generates a new version of the **smb.conf** file. To have the Samba server read these changes, you then have to restart it. You can do this by clicking the Restart SMB button on the Status page.

Configuring Samba Access from Windows

To set up a connection for a Windows client, you need to specify the Windows workgroup name and configure the password. The workgroup name is the name that appears in the My Network Places on Windows 2000, NT, and XP (the Entire Network window in the Network Neighborhood on earlier Windows versions). On Vista, this is simply called network. To set the workgroup name on Windows XP, open System on the Control Panel, and on the Computer Name panel, click the Change button for the Rename Or Change Domain Entry. This opens a dialog window with a setting for the Workgroup, where you can enter the workgroup name. The default may be WORKGROUP or MSHOME. You can set up your own workgroup name, but all your computers would have to be configured to use that name.

On your Fedora Samba server, you specify the network name in the Sever Settings window on system-config-samba. Alternatively, you can manually enter it in the **smb.conf** file, you specify the workgroup name in the **workgroup=** entry in the **global** section. The workgroup name should be uppercase and contain no spaces. The default name used on Windows XP systems is simple WORKGROUP. The **smb.conf workgroup** entry would then look like this:

```
workgroup = WORKGROUP
```

Accessing Samba Shares from Windows

On a Windows client, you see the Samba server listed when you select View Workgroups Computers from My Network Places (network on Vista). On older Windows versions, use the Entire Network folder in your Network Neighborhood. The Samba server will have as a name the description you gave it in your Samba configuration. Opening the icon will display a window with all the configured shares and printers on that Samba server.

When a Windows user wants to access a new share on the Linux system, they open their My Network Places (network on Vista) and then "Add a network place" to add a network place entry for the share, or View workgroup computers to see computers on your Windows network. Selecting the Linux Samba server will display your Samba shares. To access the share, the user will be required to enter in the user name and the Samba password. You have the option of having the username and password remembered for automatic access.

You will also need to make sure that your Windows system has enabled TCP/IP networking. This may already be the case if your Windows client is connected to a Microsoft network. If you need to connect a Windows system directly to a TCP/IP network that your Linux Samba server is running on, you should check that TCP/IP networking is enabled on that Windows system. This involves making sure that the Microsoft Network client and the TCP/IP protocol are installed, and that your network interface card (NIC adapter) is configured to use TCP/IP. The procedures differ slightly on Windows 2000 and XP, and Windows 95, 98, and ME.

Sharing Windows Directories and Printers with Samba Clients

To manage directory shares, open the Computer Management tool in the Administrative window in the Control Panel. Click Shared Folders and there you can see the Shares, Sessions, and Open folders. To add a new share, click the Shares folder and then click the Action menu and select New File Share. The Sessions and Open folders' Action menus let you disconnect active sessions and folders.

Sharing Windows Directories

To share a directory, right-click the directory and select Sharing from the pop-up menu (Sharing And Security on Windows XP). Click Share This Folder and then enter the share name, the name by which the directory will be known by Samba. You can specify whether you want to allow others to change files on the share. You can also specify a user limit (maximum allowed is the default). You can further click the Permissions button to control access by users. Here, you can specify which users will have access, as well as the type of access. For example, you could allow only read access to the directory.

Sharing Windows Printers

To share a printer, locate the printer in the Printers window and right-click it, selecting the Sharing As option. This opens the Sharing panel, where you can click the Shared As button and enter the name under which the printer will be known by other hosts. For example, on the Windows client named lizard, to have a printer called Epson Stylus Color shared as myepson, the Sharing panel for this printer would have the Shared As button selected and the name myepson entered. Then when the user double-clicks the lizard icon in the Computers Near Me window, the printer icon labeled myepson will appear.

For a Linux system to use this printer, it will have be first configured as a remote Windows printer on that Linux system. You can do this easily with the s**ystem-config-printer** tool (see Chapter 24).

User Level Security

Samba primarily provides user level security, requiring users on remote systems to login using Samba registered passwords. Samba still provides share and server level access, but these methods have been deprecated and are not recommended. User level security requires the use of Windows encrypted passwords. Windows uses its own methods of encryption. For Samba to handle such passwords, it has to maintain its own Windows compatible password database. It cannot use the Linux password databases. Windows also uses additional information for the login process like where the user logged in.

User level security requires that each user that wants to login to a Samba share from a Windows system has to have a corresponding user account on the Samba server. These are the users listed in the system-config-samba Samba Users window (see Figure 26-4). In addition, for this account, has to have a separate Samba password with which to login to the Samba share. In effect they become Samba users.

The account on the Samba server does not have to have the same name as the one on the Windows system. A Windows user name can be specified for a Samba user. On system-config-samba, the Create New Samba User window lets you enter a Windows user name in the Windows Username entry (see Figure 26-5). This mapping of windows users to Samba (Linux) users is listed in the **/etc/smbusers file**. The following maps the Windows user **rpetersen** to the Samba (Linux) user **richard**.

```
richard = rpetersen
```

When the Windows user in Windows tries to access the Samba share, the user will be prompt to login. The Windows user would then enter **rpetersen** as the user name and the Samba password that was set up for **richard**. On system-config-samba, this is the Samba password entered in the Samba Password entries in the Create New Samba Users window (see Figure 26-5)

User level security is managed by password backend databases. By default the **tdbsam** back end database is used. This is a **tdb** database file (trivial data base) that stores Samba passwords along with Windows extended information. The tdbsam database is designed for small networks. For systems using LDAP to manage users, you can use the LDAP enabled back end, **ldbsam**. The **ldbsam** database is designed for larger networks. The **smbpasswd** file previously used, is still available, but included only for backward compatibility. The default configuration entries for user access in the **smb.conf file** are shown here.

```
security = user
passdb backend = tdbsam
```

The **username map** option specifies the file used to associate Windows and Linux users. Windows users can use the Windows user name to login as the associated user. The username map file is usually **/etc/samba/smbusers**.

```
username map = /etc/samba/smbusers
```

If you are using an LDAP enabled Samba database, ldbsam, you would use special LDAP Samba tools to manage users. These are provided in the smbldap-tools package. They are prefixed with the term smbldap. There are tools for adding, modifying, and deleting users and groups like **smbldap-useradd**, **smbldap-userdelete**, and **smbldap-groupmod**. You use the **smbldap-passwd** command to manage Samba passwords with LDAP. The **smdbldap-userinfo** to obtain information about a user. You configure your LDAP Samba tools support using the **/etc/smbldap-tools/smbldap.conf** file.

Samba also provides its own Samba password PAM module, **pam_smbpass.so**. With this module, you provide PAM authentication support for Samba passwords, enabling the use of Windows hosts on a PAM-controlled network. The module could be used for authentication and password management in your PAM **samba** file. The following entries in the PAM **samba** file would implement PAM authentication and passwords using the Samba password database:

```
auth required pam_smbpass.so nodelay
password required pam_smbpass.so nodelay
```

Be sure to enable PAM in the **smb.conf** file:

```
obey pam restrictions = yes
```

smbpasswd

With user-level security, access to Samba server resources by a Windows client is allowed only to users on that client. The User name and Samba password used to access the Samba server has to be registered in the Samba password database.

Note: If you are using the older smbpasswd file, you can use the **`mksmbpasswd.sh`** script to generate an smbpasswd file made up of all the users listed in your **/etc/passwd** file. You pipe the contents of the passwd file to **`mksmbpasswd.sh`** and then use redirection (>) to create the file.

To manage users you can also use the system-config-samba, **smbpasswd** command or the **pdbedit** tool. On system-config-samba you use the Samba Users window (Preferences | Samba Users) to add or edit passwords (see Figure 26-4). The **smbpasswd** command will work on both the older smbpasswd files and the backend databases like **tdbsam**. The **pdbedit** command works only on the backend databases.

Alternatively, you can use the **smbpasswd** command in a terminal window to add, or later change, passwords. The **smbpasswd** command with the **-a** option with add a user and the **-x** option will remove one. To enable or disable users you would use the **-e** and **-d** options.

```
smbpasswd -a aleina
```

Users can use **smbpasswd** to change their own password. The following example shows how you would use **smbpasswd** to change your Samba password. If the user has no Samba password, that user can just press the ENTER key.

```
smbpasswd
Old SMB password: old-password
New SMB Password: new-password
Repeat New SMB Password: new-password
```

Should you want to use no passwords, you can use smbpasswd with the **-n** option. The **smb.conf** file will have to have the **null passwords** option set to yes. Note: If you are using the older smbpasswords file, be sure that Samba is configured to use encrypted passwords. Set the **encrypt passwords** option to **yes** and specify the SMB password file.

pdbedit

pdbedit is a command line tool with options for adding and removing users, as well as features like changing password and setting the home directory. You can also import or export the user entries to or from other backend databases. It operates just on the backend database files like **tdbsam**, not on the older **smbpasswd** files.

To add a user you would use the **-a** option and to remove a user you use the **-x** option.

```
pdbedit -a larisa
```

The **pdbedit** command lets you display more information about users. To display users from the backend database you could use the **-L** option. Add the **-v** option for detailed information. For a particular user add the user name.

```
pdbedit -Lv richard
```

For domain policies like minimum password length or retries, you use the -P option.

```
pdbedit -P
```

You use the **-i** and **-e** options to import and export database entries. The following will import entries from the old **smbpasswd** file to the new **tdbsam** backend database.

```
pdbedit -i smbpasswd -e tdbsam
```

If your system is using LDAP enabled Samba database, then use the smbldap tools to manage users and groups.

The Samba smb.conf Configuration File

Samba configuration is held in the **smb.conf** file located in the **/etc/samba** directory. Samba configuration tools like system-config-samba and SWAT, will maintain this file for you. Alternatively, you can manually edit the file directly, creating your own Samba configuration. You may have to do this if your Samba configuration proves to be very complex. Direct editing can provide more refined control over your shares.

You use the testparm command to check the syntax of any changes you have made to the **/etc/samba/smb.conf** file. Run the following in a terminal window.

```
testparm
```

The file is separated into two basic parts: one for global options and the other for shared services. A shared service, also known as *shares,* can either be file space services (used by clients as an extension of their native file systems) or printable services (used by clients to access print services on the host running the server). The file space service is a directory to which clients are given access; they can use the space in it as an extension of their local file system. A printable service provides access by clients to print services, such as printers managed by the Samba server.

The **/etc/samba/smb.conf** file holds the configuration for the various shared resources, as well as global options that apply to all resources. Linux installs an **smb.conf** file in your **/etc/samba** directory. The file contains default settings used for your distribution. You can edit the file to customize your configuration to your own needs. Many entries are commented with either a semicolon or a **#** sign, and you can remove the initial comment symbol to make them effective. For a complete listing of the Samba configuration parameters, check the Man page for **smb.conf**. An extensive set of sample **smb.conf** files is located in the **/usr/share/doc/samba*** directory in the **examples** subdirectory.

In the **smb.conf** file, global options are set first, followed by each shared resource's configuration. The basic organizing component of the **smb.conf** file is called a *section.* Each resource has its own section that holds its service name and definitions of its attributes. Even global options are placed in a section of their own, labeled **global**. For example, each section for a file space share consists of the directory and the access rights allowed to users of the file space. The section of each share is labeled with the name of the shared resource. Special sections, called **printers** and **homes**, provide default descriptions for user directories and printers accessible on the Samba server. Following the special sections, sections are entered for specific services, namely access to specific directories or printers.

A section begins with a section label consisting of the name of the shared resource encased in brackets. Other than the special sections, the section label can be any name you want to give it. Following the section label, on separate lines, different parameters for this service are entered. The parameters define the access rights to be granted to the user of the service. For example, for a directory, you may want it to be browsable, but read-only, and to use a certain printer. Parameters are entered in the format *parameter name = value.* You can enter a comment by placing a semicolon at the beginning of the comment line.

A simple example of a section configuration follows. The section label is encased in brackets and followed by two parameter entries. The **path** parameter specifies the directory to which access is allowed. The **writeable** parameter specifies whether the user has write access to this directory and its file space.

```
[mysection]
 path = /home/chris
 writeable = true
```

A printer service has the same format but requires certain other parameters. The path parameter specifies the location of the printer spool directory. The **read-only** and **printable** parameters are set to **true**, indicating the service is read-only and printable. **public** indicates anyone can access it.

```
[myprinter]
 path = /var/spool/samba
 read only = true
 printable = true
 public = true
```

Parameter entries are often synonymous but different entries that have the same meaning. For example, **`read only = no`**, **`writeable = yes`**, and **`write ok = yes`** all mean the same thing, providing write access to the user.

Hint: The **writeable** option is an alias for the inverse of the **read only** option. The **writeable = yes** entry is the same as **read only = no** entry.

Global Section

The Global section determines configuration for the entire server, as well as specifying default entries to be used in the home and directory segments. In this section, you find entries for the workgroup name, password configuration, and directory settings. Several of the more important entries are discussed here. .

The Workgroup entry specifies the workgroup name you want to give to your network. This is the workgroup name that appears on the Windows client's Network window. The default Workgroup entry in the **smb.conf** file is shown here:

```
[global]

# workgroup = NT-Domain-Name or Workgroup-Name
 workgroup = WORKGROUP
```

The workgroup name has to be the same for each Windows client that the Samba server supports. On a Windows client, the workgroup name is usually found on the Network Identification or General panel in the System tool located in the Control Panel window. On many clients, this is defaulted to WORKGROUP. If you want to keep this name, you have to change the Workgroup entry in the **smb.conf** file accordingly. The Workgroup entry and the workgroup name on each Windows client have to be the same. In this example the workgroup name is **mygroup**.

```
workgroup = mygroup
```

The server string entry, shown here, holds the descriptive name you want displayed for the server on the client systems. On Windows systems, this is the name displayed on the Samba server icon. The default is Samba Server, but you can change this to any name you want.

```
# server string is the equivalent of the NT Description field
    server string = %h server (Samba, Fedora)
```

Note: You can also configure Samba to be a Primary Domain Controller (PDC) for Windows NT networks. As a PDC, Samba sets up the Windows domain that other systems will use, instead of participating in an already established workgroup.

Name service resolution is normally provided by the WINS server (Windows NetBIOS Name Service, nmbd), which is started by the **samba** init script. If your local network already has a WINS server, you can specify that instead. The commented default entry is shown here. Replace w.x.y.z with your network's WINS server name.

```
;   wins server = w.x.y.z
```

WINS server support by your Samba **nmbd** server would have to be turned off to avoid conflicts, turning your Samba name resolution server into just a client. The commented entry to turn off WINDS support is shown here.

```
;   wins support = no
```

If you network also has its own DNS server that it wants to use for name resolution, you can enable that instead. By default this is turned off as shown here. Change the no to yes to allow use of your network's DNS server for Windows name resolution. Also, WINS server support would have to be turned off.

```
   dns proxy = no
```

Also name resolution can be instructed to check the **lmhosts** and **/etc/hosts** files first. The commented default entry is shown here.

```
;   name resolve order = lmhosts host wins bcast
```

Samba resources are normally accessed with either share or user-level security. On a share level, any user can access the resource without having to log in to the server. On a user level, each user has to log in, using a password. Furthermore, Windows 98, ME, NT, and XP clients use encrypted passwords for the login process. Passwords are encrypted by default and managed by the password database, as noted previously. In the following entries, the security is set to the user level, and the password database file uses **tdbsam**.

```
security = user
passdb backend = tdbsam
```

If you want share-level security, specify **share** as the security option. However this option is deprecated. User level security is considered the standard:

```
security = share
```

As a security measure, you can restrict access to SMB services to certain specified local networks. On the host's network, type the network addresses of the local networks for which you want to permit access. To deny access to everyone in a network except a few particular hosts, you can use the EXCEPT option after the network address with the IP addresses of those hosts. The localhost (127) is always automatically included. The next example allows access to two local networks:

```
hosts allow = 192.168.1. 192.168.2.
```

To enable printing, specify the load printer configurations.

```
load printers = yes
```

Other options like **printing** let you specify a different printing server (CUPS is the default).

```
printing = cups
```

You can use a guest user login to make resources available to anyone without requiring a password. A guest user login would handle any users who log in without a specific account. On Linux systems, by default Samba will use the **nobody** user as the guest user. Alternatively, you can set up and designate a specific user to use as the guest user. You designate the guest user with the Guest Account entry in the **smb.conf** file. The commented **smb.conf** file provided with Samba

currently lists a commented entry for setting up a guest user called **nobody**. You can make this the user you want to be used as the guest user. Be sure to add the guest user to the password file:

```
guest account = nobody
```

Homes Section

The Homes section specifies default controls for accessing a user home directory through the SMB protocols by remote users. Setting the **browseable** entry to **no** prevents the client from listing the files in a file browser. The **writeable** entry specifies whether users have read and write control over files in their home directories. The **create mode** and **directory mode** set default permissions for new files and directories. The **valid users** entry uses the %S macro to map to the current service.

```
[homes]
 comment = Home Directories
 browseable = no
 read only = no
 valid users = %S
 create mode = yes
 directory mask = 775
```

The printer and print$ Sections

The printers section specifies the default controls for accessing printers. These are used for printers for which no specific sections exist. Setting **browseable** to **no** simply hides the Printers section from the client, not the printers. The **path** entry specifies the location of the spool directory Samba will use for printer files. To enable printing at all, the **printable** entry must be set to yes. To allow guest users to print, set the **guest ok** entry to **yes**. This specifies the command the server actually uses to print documents. The standard implementation of the Printers section is shown here:

```
[printers]
 comment = All Printers
 path = /var/spool/samba
 browseable = no
 guest ok = yes
 writable = no
 printable = yes
 read only = yes
 create mask = 0700
```

The print$ section specifies where a Windows client can find a print driver on your Samba server. The printer drivers are located in the **/var/lib/samba/printers** directory and are read only.

```
# Windows clients look for this share name as a source of downloadable
# printer drivers
[print$]
   comment = Printer Drivers
   path = /var/lib/samba/printers
   browseable = yes
   read only = yes
   guest ok = no
```

Shares

Sections for specific shared resources, such as directories on your system, are usually placed after the Homes and Printers sections. For a section defining a shared directory, enter a label for the system. Then, on separate lines, enter options for its pathname and the different permissions you want to set. In the **path =** *option,* specify the full pathname for the directory. The **comment =** *option* holds the label to be given the share. You can make a directory writable, public, or read-only. You can control access to the directory with the **valid users** entry. With this entry, you can list those users permitted access. For those options not set, the defaults entered in the Global, Homes, and Printers segments are used. For more examples, check those in the original **smb.conf** file. The following example is the **myprojects** share. Here the **/myprojects** directory is defined as a share resource that is open to any user with guest access.

```
[myprojects]
     comment = Great Project Ideas
     path = /myprojects
     read only = no
     guest ok = yes
```

To limit access to certain users, you can list a set of valid users. Setting the **guest ok** option to **no** closes it off from access by others.

```
[mynewmusic]
 comment =  New Music
 path = /home/specialprojects
 valid users = mark
 guest ok = no
 read only = no
```

To allow complete public access, set the **guest ok** entry to **yes**, with no valid user's entry.

```
[newdocs]
 comment =  New Documents
 path = /home/newdocs
 guest ok = yes
 read only = no
```

To set up a directory that can be shared by more than one user, where each user has control of the files he or she creates, simply list the users in the Valid Users entry. Permissions for any created files are specified in the Advanced mode by the Create Mask entry (same as create mode). In this example, the permissions are set to 765, which provides read/write/execute access to owners, read/write access to members of the group, and only read/execute access to all others (the default is 744, read-only for group and other permission):

```
[myshare]
 comment = Writer's projects
 path = /usr/local/drafts
 valid users = justin chris dylan
 guest ok = no
 read only = no
 create mask = 0765
```

Printers

Access to specific printers is defined in the Printers section of the **smb.conf** file. For a printer, you need to include the Printer and Printable entries, as well as specify the type of Printing server used. With the Printer entry, you name the printer, and by setting the Printable entry to yes, you allow it to print. You can control access to specific users with the valid users entry and by setting the Public entry to no. For public access, set the Public entry to yes. For the CUPS server, set the printing option cups.

The following example sets up a printer accessible to guest users. This opens the printer to use by any user on the network. Users need to have write access to the printer's spool directory, located in **/var/spool/samba**. Keep in mind that any printer has to first be installed on your system. The following printer was already installed as myhp. You use the CUPS administrative tool to set up printers for the CUPS server. The Printing option can be inherited from general Printers share.

```
[myhp]
      path = /var/spool/samba
      read only = no
      guest ok = yes
      printable = yes
      printer = myhp
      oplocks = no
      share modes = no
      printing = cups
```

As with shares, you can restrict printer use to certain users, denying it to public access. The following example sets up a printer accessible only by the users **larisa** and **aleina** (you could add other users if you want). Users need to have write access to the printer's spool directory.

```
[larisalaser]

      path = /var/spool/samba
      read only = no
      valid users = larisa aleina
      guest ok = no
      printable = yes
      printing = cups
      printer = larisalaser
      oplocks = no
      share modes = no
```

Variable Substitutions

For string values assigned to parameters, you can incorporate substitution operators. This provides greater flexibility in designating values that may be context-dependent, such as usernames. For example, suppose a service needs to use a separate directory for each user who logs in. The path for such directories could be specified using the **%u** variable that substitutes in the name of the current user. The string **path = /tmp/%u** would become **path = /tmp/justin** for the **justin** user and **/tmp/dylan** for the **dylan** user. Table 26-4 lists several of the more common substitution variables.

Variable	Description
%S	Name of the current service
%P	Root directory of the current service
%u	Username of the current service
%H	Home directory of the user
%h	Internet hostname on which Samba is running
%m	NetBIOS name of the client machine
%L	NetBIOS name of the server
%M	Internet name of the client machine
%I	IP address of the client machine

Table 26-4: Samba Substitution Variables

Testing the Samba Configuration

After you make your changes to the **smb.conf** file, you can then use the testparm program to see if the entries are correctly entered. testparm checks the syntax and validity of Samba entries. By default, testparm checks the **/etc/samba/smb.conf** file. If you are using a different file as your configuration file, you can specify it as an argument to testparm. You can also have testparm check to see if a particular host has access to the service set up by the configuration file.

With SWAT, the Status page will list your connections and shares. From the command line, you can use the **smbstatus** command to check on current Samba connections on your network.

To check the real-time operation of your Samba server, you can log in to a user account on the Linux system running the Samba server and connect to the server.

Samba Public Domain Controller: Samba PDC

Samba can also operate as a Public Domain Controller (PDC). The domain controller will be registered and advertised on the network as the domain controller. The PDC provides a much more centralized way to control access to Samba shares. It provides the netlogon service and a NETLOGON share. The PDC will set up machine trust accounts for each Windows and Samba client. Though you can do this manually, Samba will do it for you automatically. You can find out more about Samba PDC at:

http://us1.samba.org/samba/docs/man/Samba-HOWTO-Collection/samba-pdc.html

Microsoft Domain Security

As noted in the Samba documentation, the primary benefit of Microsoft domain security is single-sign-on (SSO). In effect, logging into your user account also logs you into access to your entire network's shared resourses. Instead of having to be separately authenticated anytime you try to access a shared network resource, you are already authenticated. Authentication is managed using Security IDs (SID) that consists of a network ID (NID) and a relative ID (RID). The RID

references your personal account. A separate RID is assigned to every account, even those for groups or system services. The SID is use to set up access control lists (ACL) the different shared resourses on your network, allowing a resource to automatically identify you.

Note:You can also enable and configure the Samga PDC controller using system-config-authentication (System | Administration | Authorization). On the Authentication tab, click the Enable SMB Support check box, and the click the Configure button. This opens a dialog where you can ente the workgroup and the domain controller URL (PDC).

Essential Samba PDC configuration options

To configure your PDC edit the Domain Control Options section in the **smb.conf** file. Here you will find entries for configuring your Samba PDC.

The essential PDC options are shown here.

```
netbios name = mysystem
workgroup = myworkgroup
domain logons = yes
domain master = yes
security = user
```

Basic configuration

The netbios name is the name you want to give to the PDC server.

```
netbios name = MYSERVER
```

Like most Samba configurations, the PDC requires a Samba backend. The **tdbsam** is already configured for you. The security level should be **user**. This is normally the default and should already be set. The **smb.conf** entries are shown here:

```
security = user
passdb backend = tdbsam
```

Domain Logon configuration

Samba PDC uses domain logons service whereby a user can log on to the network. The domain logon service is called the netlogon service by Microsoft. The samba share it uses is also called netlogon. To configure the domain logon service you set the domain logons entry to yes. The PDC must also be designated the domain master. These are commented in your smb.conf file (semi-colon, ;). Remove the semi-colon to uncomment them. The entries are already set to yes.

```
domain master = yes
domain logons = yes
```

The login script can be one set by the system or by users. The logon path references the profile used for a user.

```
# the login script name depends on the machine name
logon script = %m.bat
# the login script name depends on the unix user used
logon script = %u.bat
logon path = \\%L\Profiles\%u
```

You can then enable user add and delete operations for adding and removed users from the PDC. The add machine entry allows Samba to automatically add trusted machine accounts for Windows systems when they first join the PDC controlled network.

```
add user script = /usr/sbin/useradd "%u" -n -g users
add group script = /usr/sbin/groupadd "%g"
add machine script = /usr/sbin/useradd -n -c "Workstation (%u)" -M -d /nohome -s
/bin/false "%u"
delete user script = /usr/sbin/userdel "%u"
delete user from group script = /usr/sbin/userdel "%u" "%g"
delete group script = /usr/sbin/groupdel "%g"
```

You then need to set up a netlogon share in the **smb.conf** file. This share holds the **netlogon** scripts—in this case, the **/var/lib/samba/netlogon** directory—which should not be writable but should be accessible by all users (Guest OK). In the share definitions section of the **smb.conf** file you will find the [netlogon] section commented. Remove the semi-colon comments from the entry, as shown here.

```
# Un-comment the following and create the netlogon directory for Domain Logon
[netlogon]
comment = Network Logon Service
path = /var/lib/samba/netlogon
guest ok = yes
writable = no
share modes = no
```

You then should enable the roving profile share. Un-comment the following to provide a specific roving profile share, located just after the netlogon shares. The profile share is where user netlogon profiles are stored.

```
# the default is to use the user's home directory
[Profiles]
path = /var/lib/samba/profiles
browseable = no
guest ok = yes
```

Master Browser configuration

The PDC has browser functionality, with which it locates systems and shares on your network. Browser features are set in the Browser Control Options section of the **smb.conf**. The local master option is use only if you already have another PDC that you want to operate as the local master. Normally you would leave the entry commented, as shown here. You could have several domain controllers operating on your network. Your Microsoft network holds an election choose which should be the master. The os level set the precedence for this PDC. It should be higher than 32 to gain preference over other domain controllers on your network, insuring this PDC's election as the primary master controller. The preferred master option starts the browser election on start up.

```
;       local master = no
os level = 33
preferred master = yes
```

Note: WIndows XP Home Edition cannot be used with domain security (PDC).

Accessing Samba Services with Clients

Client systems connected to the SMB network can access the shared services provided by the Samba server. Windows clients should be able to access shared directories and services automatically through the Network Neighborhood and the Entire Network icons on a Windows desktop. For Linux systems connected to the same network, Samba services can be accessed using the GNOME Nautilus file manager and KDE file manager, as well as special Samba client programs.

With the Samba smbclient**,** a command line client, a local Linux system can connect to a shared directory on the Samba server and transfer files, as well as run shell programs. Using the **`mount`** command with the **`-t cifs`** option**,,** directories on the Samba server can be mounted to local directories on the Linux client. The **`cifs`** option invokes **mount.cifs** to mount the directory.

Accessing Windows Samba Shares from GNOME

You can use Nautilus (the GNOME file manager) to access your Samba shares. You can open the My Computer icon and then the Network icon. This will display the icons for your network. The Windows Network icon will hold the Windows workgroups that your Windows hosts are part of. Opening up the Windows Network icon will list your Windows network groups, like WORKGROUP. Opening up the Windows group icon will list the hosts in that group. These will show host icon for your shared Windows hosts. Clicking a host icon will list all the shared resources on it.

Alternatively, you can start Nautilus in browser mode and enter the **smb:** protocol to display all the Samba and Windows networks, from which you can access the Samba and Windows shares.

smbclient

The smbclient utility operates like FTP to access systems using the SMB protocols. Whereas with an FTP client you can access other FTP servers or Unix systems, with smbclient you can access SMB-shared services, either on the Samba server or on Windows systems. Many smbclient commands are similar to FTP, such as **`mget`** to transfer a file or **`del`** to delete a file. The smbclient program has several options for querying a remote system, as well as connecting to it. See the **smbclient** Man page for a complete list of options and commands. The smbclient program takes as its argument a server name and the service you want to access on that server. A double slash precedes the server name, and a single slash separates it from the service. The service can be any shared resource, such as a directory or a printer. The server name is its NetBIOS name, which may or may not be the same as its IP name. For example, to specify the **myreports** shared directory on the server named **turtle.mytrek.com**, use **//turtle.mytrek.com/myreports**. If you must specify a pathname, use backslashes for Windows files and forward slashes for Unix/Linux files:

```
//server-name/service
```

You can also supply the password for accessing the service. Enter it as an argument following the service name. If you do not supply the password, you are prompted to enter it.

You can then add several options to access shares, such as the remote username or the list of services available. With the **`-I`** option, you can specify the system using its IP address. You use the **`-U`** option and a login name for the remote login name you want to use on the remote system.

Attach `%` with the password if a password is required. With the **`-L`** option, you can obtain a list of the services provided on a server, such as shared directories or printers. The following command will list the shares available on the host **turtle.mytrek.com**:

```
smbclient -L turtle.mytrek.com
```

To access a particular directory on a remote system, enter the directory as an argument to the **`smbclient`** command, followed by any options. For Windows files, you use backslashes for the pathnames, and for Unix/Linux files, you use forward slashes. Once connected, an SMB prompt is displayed and you can use smbclient commands such as **`get`** and **`put`** to transfer files. The **`quit`** or **`exit`** commands quit the smbclient program. In the following example, smbclient accesses the directory **myreports** on the **turtle.mytrek.com** system, using the **dylan** login name:

```
smbclient //turtle.mytrek.com/myreports -I 192.168.0.1 -U dylan
```

In most cases, you can simply use the server name to reference the server, as shown here:

```
smbclient //turtle.mytrek.com/myreports -U dylan
```

If you are accessing the home directory of a particular account on the Samba server, you can simply specify the **homes** service. In the next example, the user accesses the home directory of the **aleina** account on the Samba server, after being prompted to enter that account's password:

```
smbclient //turtle.mytrek.com/homes -U aleina
```

You can also use smbclient to access shared resources located on Windows clients. Specify the computer name of the Windows client along with its shared folder. In the next example, the user accesses the **windata** folder on the Windows client named **lizard**. The folder is configured to allow access by anyone, so the user just presses the ENTER key at the password prompt.

```
smbclient //lizard/windata
```

Once logged in, you can execute smbclient commands to manage files and change directories. Shell commands can be executed with the **`!`** operator. To transfer files, you can use the **`mget`** and **`mput`** commands, much as they are used in the FTP program. The **`recurse`** command enables you to turn on recursion to copy whole subdirectories at a time. You can use file-matching operators, referred to here as *masks,* to select a certain collection of files. The file-matching (mask) operators are **`*`**, **`[]`**, and **`?`**. The default mask is **`*`**, which matches everything. The following example uses **`mget`** to copy all files with a **.c** suffix, as in **myprog.c**:

```
smb> mget *.c
```

mount.cifs: mount -t cifs

Using the **`mount`** command with the **`-t cifs`** option., a Linux or Unix client can mount a shared directory onto its local system. The **`cifs`** option invokes the **`mount.cifs`** command to perform the mount operation. The syntax for the **`mount.cifs`** command is similar to that for the **`smbclient`** command, with many corresponding options. The **`mount.cifs`** command takes as its arguments the Samba server and shared directory, followed by the local directory where you want to mount the directory. The following example mounts the **myreports** directory onto the **/mnt/myreps** directory on the local system:

Instead of using **mount.cifs** explicitly, you use the **mount** command with the file system type **cifs**. **mount** will then run the **/sbin/mount.cifs** command, which will invoke **smbclient** to mount the file system:

```
mount -t cifs //turtle.mytrek.com/myreports /mnt/myreps -U dylan
```

To unmount the directory, use the **cifs.umount** command with the local directory name, as shown here:

```
cifs.umount /mnt/myreps
```

To mount the home directory of a particular user on the server, specify the **homes** service and the user's login name. The following example mounts the home directory of the user **larisa** to the **/home/chris/larisastuff** directory on the local system:

```
mount -t samba //turtle.mytrek.com/homes /home/chris/larisastuff -U larisa
```

You can also mount shared folders on Windows clients. Just specify the computer name of the Windows client along with its folder. If the folder name contains spaces, enclose it in single quotes. In the following example, the user mounts the **windata** folder on **lizard** as the **/mylinux** directory. For a folder with access to anyone, just press ENTER at the password prompt:

```
$ mount -t cifs //lizard/windata  /mylinux
Password:
$ ls /mylinux
_hi_mynewdoc.doc_myreport.txt
```

To unmount the shared folder when you are finished with it, use the **cifs.umount** command.

```
umount /mylinux
```

You could also specify a username and password as options, if user-level access is required:

```
mount -t cifs -o username=chris passwd=mypass //lizard/windata /mylinux
```

You can also use the cifs type in an **/etc/fstab** entry to have a Samba file system mounted automatically:

```
//lizard/windata /mylinux cifs defaults 0 0
```

fedora

27. Distributed Network File Systems

For very large distributed systems like Linux clusters, Linux also supports distributed network file systems, such as Oracle Cluster File System for Linux (OCFS2), Intermezzo, and Red Hat Global File System (GFS and GFS 2). These systems build on the basic concept of NFS as well as RAID techniques to create a file system implemented on multiple hosts across a large network, in effect, distributing the same file system among different hosts at a very low level (see Table 27-1). You can think of it as a kind of RAID array implemented across network hosts instead of just a single system. Instead of each host relying on its own file systems on its own hard drive, they all share the same distributed file system that uses hard drives collected on different distributed servers. This provides far more efficient use of storage available to the hosts, as well as providing for more centralized management of file system use.

Website	Name
www.fedoraproject.org/wiki/Tools/GFS	Fedora GFS resources and links
http://oss.oracle.com/projects/ocfs2/	OCFS2, Oracle Cluster File System for Linux
http://sources.redhat.com/cluster/gfs/	Global File System
www.sun.com/software/products/lustre/	Lustre

Table 27-1: Distributed File Systems

Note: The Parallel Virtual File System (PVFS) implements a distributed network file system using a management server that manages the files system on different I/O servers. Management servers maintain the file system information, including access permissions, directory structure, and metadata information, www.pvfs.org.

Red Hat Global File System (GFS and GFS 2)

Red Hat has released its Global File System (GFS) as an open source freely available distributed network file system. The original GFS version has been replaced with the new version of GFS, GFS 2, which uses a similar set of configuration and management tools, as well as native kernel support. Instead of a variety of seemingly unrelated packages, GFS 2 is implemented with just three: **gfs2-utils**, **cman**, and **lvm2-cluster**. Native kernel support for GFS 2 provides much of the kernel-level operations.

To configure GFS you can use **system-config-cluster**, the Red Hat GUI configuration tool for GFS. Access it from Administration | System | Server Settings | Cluster Management.

A distributed network file system builds on the basic concept of NFS as well as RAID techniques to create a file system implemented on multiple hosts across a large network, in effect, distributing the same file system among different hosts at a very low level. You can think of it as a kind of RAID array implemented across network hosts instead of just a single system. That is, instead of each host relying on its own file systems on its own hard drive, they all share the same distributed file system that uses hard drives collected on different distributed servers. This provides far greater efficient use of storage available to the hosts and provides for more centralized management of file system use. GFS can be run either directly connected to a SAN (storage area network) or using GNBD (Global Network Block Device) storage connected over a LAN. The best performance is obtained from a SAN connection, whereas a GNBD format can be implemented

easily using the storage on LAN (Ethernet)–connected systems. As with RAID devices, mirroring, failover, and redundancy can help protect and recover data.

GFS separates the physical implementation from the logical format. A GFS appears as a set of logical volumes on one seamless logical device that can be mounted easily to any directory on your Linux file system. The logical volumes are created and managed by the Cluster Logical Volume Manager (CLVM), which is a cluster-enabled LVM. Physically, the file system is constructed from different storage resources, known as cluster nodes, distributed across your network. The administrator manages these nodes, providing needed mirroring or storage expansion. Should a node fail, GFS can fence a system off until it has recovered the node. Setting up a GFS requires planning. You have to determine ahead of time different settings like the number and names of your Global File Systems, the nodes that will be able to mount the file systems, fencing methods, and the partitions and disks to use.

For detailed information check the Cluster Project Page site at **www.sourceware.org/cluster**. Listed are the packages used in both GFS (Cluster Components—Old) and GFS 2 (Cluster Components—New). Here you will find links for documentation like the clustering FAQ. The Red Hat GFS Administrators Guide can be helpful but may be dated. The guide can be found on the Red Hat documentation page located at **www.redhat.com**. (Bear in mind that GFS now uses logical volumes instead of pools to set up physical volumes.)

Website	Name
http://redhat.com/software/gfs	Global File System (Red Hat commercial version)
http://redhat.com/docs/manuals/csgfs/	Global File System Red Hat manuals (Red Hat Enterprise implementation)
http://sourceware.org/cluster	Cluster Project website, which includes links for GFS documentation
/etc/cluster/cluster.conf	GFS cluster configuration file (css)
system-config-cluster	Red Hat GUI configuration tool for GFS, Administration | System | Server Settings | Cluster Management

GFS 2 Packages

The original GFS, GFS 1, used a variety of separate packages for cluster servers and management tools, which did not appear related just by their names. With GFS 2, these packages have been combined into the **cman** and **gfs2-tools** packages. Here you will find tools such as fence, cman cluster manager, dlm locking control, and **ccs** cluster configuration. Cluster configuration is supported by the Cluster Configuration System, **ccs**. Fencing is used to isolate failed resources. It is supported by in the **fence** server. LVM cluster support is located in a separate package, **lvm2-cluster** .

To run a cluster, you need both a cluster manager and locking mechanism. **cman** with the Distributed Lock Manager (**dlm**) implements cluster management and locking. **cman** manages connections between cluster devices and services, using **dlm** to provide locking. The **dlm** locking mechanism operates as a daemon with supporting libraries.

All these services are invoked by the **cman** script, which checks the **/etc/cluster/cluster.conf** file for cluster configuration.

GFS 2 Service Scripts

To start the GFS file system, you run the **cman** script to start the needed daemon and implement your configuration. The **cman** script will run the **ccsd** to start up configuration detection, **fenced** for fencing support, **dlm_controld** for cluster management **dlm** locking, **gfs_controld** for cluster management mount and recovery events, and **cman** for cluster management. The script will check for any GFS configuration settings in **/etc/cluster/cluster.conf**. You the use the **gfs2-tools** script to mount your GFS 2 file systems. To shut down the GFS file system service, you use the **cman** script with the **stop** option.

```
/etc/init.d/cman start
/etc/init.d/gfs2-tools start
```

The **gfs2** service script will mount GFS file systems to the locations specified in the **/etc/fstab** file. You will need entries for all the GFS file systems you want to mount in **/etc/fstab**. The stop option will unmount the file systems. The **gfs2** script can be started automatically with the Services tool (System | Administration | Services); select **gfs2**.

Implementing a GFS 2 File System

To set up a GFS 2 file system, you first need to create cluster devices using the physical volumes and organizing them into logical volumes. You use the CLVM (Clustering Logical Volume Manager) to set up logical volumes from physical partitions (in the past you used a volume manager called pool to do this). You can then install GFS file systems on these logical volumes directly. CLVM operates like LVM, using the same commands. It works over a distributed network and requires that the **clvmd** server be running (**lvm2-cluster** package).

You then configure your system with the Cluster Configuration System. Create a **/etc/cluster/cluster.conf** file and set up your configuration. The configuration will include information like the nodes used, the fencing methods, and the locking method used. Consult the **cluster.conf** Man page for configuration details. Test the configuration with the **ccs_test** tool.

```
ccs_test mygfs
```

You then use the **ccs_tool** to create **cluster.ccs**, **fence.ccs**, and **node.ccs** configuration files. These files are organized into a CCS archive that is placed on each node and cluster device.

On each node, start the **ccsd** configuration, the **fenced** fencing server, and the locking method you want to use, such as **dlm**. Check the respective Man pages for details on the locking servers. You can start the servers with their service scripts as noted previously.

To create new file systems on the cluster devices, you use the **gfs2_mkfs** command and mount them with the **-t gfs2** option. The following command creates a GFS file system on the **/dev/gv0/mgfs** and then mounts it to the **/mygfs** directory. For **gfs2_mkfs**, the **-t** option indicates the lock table used and the **-p** option specifies the lock protocol. The **-j** option specifies the number of journals.

Command	Description
ccs_tool	CCS configuration update tool
ccs_test	CCS diagnostic tool to test CCS configuration files
ccsd	Daemon run on nodes to provide CCS configuration data to cluster software
clvmd	Cluster Logical Volume Manager daemon, needed to create and manage LVM cluster devices, also a service script to start **clvmd**, **lvm2-cluster** package.
cman	The Cluster Manager, **cman**, startup script, uses dlm for locking (**cman** is run as a kernel module directly)
cman_tool	Manages cluster nodes, requires **cman**
dlm	Distributed Lock Manager, implemented as a kernel module, invoked by the **cman** script
fence	Fence overview
fenced	Fencing daemon, also a service script for starting the fenced daemon
fence_tool	Manages the **fenced** daemon
fence_node	Invokes a fence agent
fencing agents	Numerous fencing agents available for different kinds of connections, see **fence** Man page
fence_manual	Fence agent for manual interaction
fence_ack_manual	User interface for fence_manual
gfs_controld	Manages mount, unmount, recovery, and locking events for GFS.
gfs2	GFS 2 service script to mount GFS 2 file systems, also a Man page overview
gfs2_mount	Invoked by mount; use **-t gfs2** mount option
gfs2_fsck	The GFS 2 file system checker
gfs2_grow	Grows a GFS 2 file system
gfs2_jadd	Adds a journal to a GFS 2 file system
mkfs.gfs2	Makes a GFS 2 file system
gfs2_quota	Manipulates GFS 2 disk quotas
gfs2_tool	Manages a GFS 2 file system
getfacl	Gets the ACL permissions for a file or directory
setfacl	Sets access control (ACL) for a file or directory
rmanager	Resource Group Manager, manage user services

Table 27-2: GFS Tools, Daemons, and Service Scripts

```
gfs2_mkfs  -t mycluster:mygfs  -p lock_dlm  -j 2   /dev/vg0/mgfs
mount -t gfs /dev/vg0/mgfs /gfs1
```

To have the **gfs** service script mount the GFS file system for you, you need to place an entry for it in the **/etc/fstab** file. If you do not want the file system automatically mounted, add the **noauto** option.

```
/dev/vg0/mgfs   /mygfs    gfs2   noauto,defaults   0   0
```

With GFS **/etc/fstab** entries, you can then use the **gfs2** script to mount the GFS file system.

```
/etc/int.d/gfs2-tools start
```

GFS Tools

GFS has several commands in different categories, such as those that deal with fencing, like **fence_tool**; **gulm_tool** to manage gulm locking; and those used for configuration, like **cman_tool**. The GFS commands for managing GFS file systems are listed in Table 27-2. Check their respective Man pages for detailed descriptions.

GFS File System Operations

. Several GFS commands manage the file system, such as **gfs2_mount** for mounting file systems, **gfs2_mkfs** to make a GFS file system, **gfs2_fsck** to check and repair, and **gfs2_grow** to expand a file system. Check their respective Man pages for detailed descriptions.

Note: For GFS 1, you use the same names for the GFS tools without the number 2; that is, **gfs** instead of **gfs2**.

To mount a GFS file system, you use the **mount** command specifying **gfs2** as the mount type, as in

```
mount  -t gfs2  /dev/vg0/mgfs  /mygfs
```

This will invoke the **gfs2_mount** tool to perform the mount operation. Several GFS-specific mount options are also available, specified with the **-o** option, such as **lockproto** to specify a different lock protocol and **acl** to enable ACL support .

To check the status of a file system, you can use **gfs2_fsck**. This tool operates much like fsck, checking for corrupt systems and attempting repairs. You must first unmount the file system before you can use **gfs2_fsck** on it.

Should you add available space to the device on which a GFS file system resides, you can use **gfs2_grow** to expand the file system to that available space. It can be run on just one node to expand the entire cluster. If you want journaling, you first have to add journal files with the **gfs2_jadd** tool. **gfs2_grow** can only be run on a mounted GFS file system.

Journal files for GFS are installed in space outside of the GFS file system, but on the same device. After creating a GFS file system, you can run **gfs2_add** to add the journal files for it. If you are expanding a current GFS file system, you need to run **gfs2_add** first. Like **gfs2_grow**, **gfs2_add** can only be run on mounted file systems. With the **setfacl** command you can set permissions for files and directories.

As noted previously, to create a GFS file system you use the **gfs2_mkfs** command. The **-t** option specifies the lock table to use, the **-j** options indicates the number of journals to create, and the **-p** option specifies the lock protocol to use.

The Resource Group Manager, **rgmanager**, provides a command line interface for managing user services and resources on a GFS file system, letting you perform basic administrative tasks like setting user quotas, shutting down the system (**clushutdown**), and getting statistics on GFS use (**clustat**). The primary administrative tool is **clusterfs**. Options can be set in the **/etc/sysconfig/cluster** file. You start up **rgmanager** with the **rgmanager** script. This starts up the **clurgmgrd** daemon, providing access to the GFS system.

GFS also supports access controls. You can restrict access by users or groups to certain files or directories, specifying read or write permissions. With the **setfacl** command you can set permissions for files and directories. You use the **-m** option to modify an ACL permission and **-x** to delete it. The **getfacl** obtains the current permissions for file or directory. The following sets read access by the user **dylan** to **myfile**.

```
setfacl -m u:dylan:r myfile
```

GFS 1

GFS 1, the first version of GFS, implements GFS with a series of separate packages, seemingly unrelated. To use GFS 1, you have to install a number of different packages that include the GFS 1 tools, the locking method you want to use, and configuration tools. The GFS 1 tools and locking methods also have several corresponding kernel module and header packages. See **www.sourceware.org/cluster** under the heading "Cluster Components—Old" for a listing of GFS 1 tools. There were kernel module packages for the different types of kernels: i586, i686, SMP, and Xen.

Part 6: Network Support

fedora

28. Network Connections

With Fedora 10, network configuration is now managed primarily by the Network Manager, though the standard tools from previous releases are still used. Network configuration differs depending on whether you are connected to a local area network (LAN) with an Ethernet card or are using a DSL or ISDN modem, a wireless connection, or a dial-up modem connection. You had the opportunity to enter your LAN network settings during the installation process. For modifying your LAN settings and for configuring other kinds of interfaces such as DSL, wireless, or ISDN connections, you can configure your network connection using system-config-network, as well as add a new connection. Table 28-1 lists several different network configuration tools.

Network Configuration Tool	Description
Network Manager	Automates wireless and standard network connection selection and notification.
`system-config-network`	Fedora network configuration tool for all types of connections.
Network Manager	Automates wireless and standard network connection selection and notification.
KNetworkManager	KDE tool to automate wireless and standard network connection selection and notification.
`system-control-network`	Fedora network device control tool for turning network connections on an off.
`system-config-services`	Starts and stops servers, including network servers (smb for Samba, httpd for Web, bind for DNS, and nfs for NFS).
`system-config-firewall`	Sets up a network firewall.
`system-config-bind`	Configures a domain name server.
`system-config-samba`	Configures Samba shares.
`system-config-nfs`	Configures NFS shares.
`system-config-httpd`	Configures an Apache Web server.
`system-config-netboot`	Configures diskless workstations and network installation.
`wvdial`	PPP modem connection, enter on a command line.
`pand`	Implements the Bluetooth Personal Network.

Table 28-1: Fedora Network Configuration Tools

Network Information: Dynamic and Static

If you are on a network, you may need to obtain certain information to configure your interface. Most networks now support dynamic configuration using either the older Dynamic Host Configuration Protocol (DHCP) or the new IPv6 Protocol and its automatic address configuration. In this case, you need only check the DHCP entry in most network configuration tools. For IPv6, you would check the Enable IPv6 configuration entry in the system-config-network device configuration window (see Figure 28-4 later in this chapter). However, if your network does not

support DHCP or IPv6 automatic addressing, you will have to provide detailed information about your connection. Such connections are known as static connections, whereas DCHP and IPv6 connections are dynamic. In a static connection, you need to manually enter your connection information such as your IP address and DNS servers, whereas in a dynamic connection this information is automatically provided to your system by a DHCP server or generated by IPv6 when you connect to the network. For DHCP, a DHCP client on each host will obtain the information from a DHCP server serving that network. IPv6 generates its addresses directly from the device and router information such as the device hardware MAC address.

In addition, if you are using a DSL dynamic, ISDN, or modem connection, you will also have to supply provider, login, and password information, whether your system is dynamic or static. You may also need to supply specialized information such as DSL or modem compression methods, dialup number, or wireless channels to select.

You can obtain most of your static network information from your network administrator or from your ISP (Internet service provider). You would need the following information:

The device name for your network interface For LAN and wireless connections, this is usually an Ethernet card with the name **eth0** or **eth1**. For a modem, DSL, or ISDN connection, this is a PPP device named **ppp0** (**ippp0** for ISDN). Virtual private network (VPN) connections are also supported.

Hostname Your computer will be identified by this name on the Internet. Do not use localhost; that name is reserved for special use by your system. The name of the host should be a simple word, which can include numbers, but not punctuation such as periods and backslashes. The hostname includes both the name of the host and its domain. For example, a hostname for a machine could be **turtle**, whose domain is **mytrek.com**, giving it a hostname of **turtle.mytrek.com**.

Domain name This is the name of your network.

The Internet Protocol (IP) address assigned to your machine This is needed only for static Internet connections. Dynamic connections use the DHCP protocol to automatically assign an IP address for you. Every host on the Internet is assigned an IP address. Traditionally, this address used an IPv4 format consisting of a set of four numbers, separated by periods, which uniquely identifies a single location on the Internet, allowing information from other locations to reach that computer. Networks are now converting to the new IP protocol version 6, IPv6, which uses a new format with a much more complex numbering sequence.

Your network IP address Static connections only. This address is usually similar to the IP address, but with one or more zeros at the end.

The netmask Static connections only. This is usually 255.255.255.0 for most networks. If, however, you are part of a large network, check with your network administrator or ISP.

The broadcast address for your network, if available (optional) Static connections only. Usually, your broadcast address is the same as your IP address with the number 255 added at the end.

The IP address of your network's gateway computer Static connections only. This is the computer that connects your local network to a larger one like the Internet.

Name servers Static connections only. The IP address of the name servers your network uses. These enable the use of URLs.

NIS domain and IP address for an NIS server Necessary if your network uses an NIS server (optional).

User login and password information Needed for dynamic DSL, ISDN, and modem connections.

Static IP Address Configuration

Most servers use static IP addresses, making them located easily on the Internet or local network. Unless specifically configured during installation, Fedora does not configure static IP addresses automatically. Instead the host name is associated with the localhost, and the IP address is determined automatically by a network DHCP server. If your server host is using a static IP address, you will have to manually configure your network to use, if you have not done so already.

To configure a static IP address, you can use the Hosts pane in the system-config-network tool. Click the New button and enter the host name and its IP address. You can also directly edit the **/etc/hosts** file and add a line for your static IP address and its host name.

Network Manager can interfere with a static IP address configuration. Network Manager will always try to locate any available network address from an active DHCP server or Wireless router. You may have to turn off Network manager. Use system-config-services to turn off Network Manager. Select Network Manager and click disable. To turn it off for other runlevels, click the Customize button and uncheck those runlevels. This will prevent Network Manager from starting up at those runlevels.

You may want to run your server at runlevel 3, but use runlevel 5 to configure it. Runlevel 5 runs the X server which can degrade efficiency for busy server. At runlevel 3, you are running just the command line interface (Shell), with not X server. This runlevel would be more appropriate for servers. For a server with a static IP address, you could turn all Network Manager just for runlevel 3 (Customize window), but leave it on for runlevel 5 (desktop and X server), allowing for other kinds of network connections.

User network configuration: Network Manager, System | Preferences | Network Configuration

Network Manager will automatically detect your network connections, both wired and wireless. Network Manager makes use the automatic device detection capabilities of udev and HAL to configure your connections. Should you instead need to configure your network connections manually, you still use Network Manager, selecting the manual options and entering the required network connection information.

Network Manager is user specific. When a user logs in, it selects the network connection preferred by that user. The first time a user runs NetworkManager, the notification applet will display a list of current possible connections. The user can then choose one. For wired connections, a connection can be started automatically, when the system starts up. Initial settings will be supplied from system-wide configuration.

Fedora 10 uses an enhanced version of Network Manager which can also manually configure any network connection. To configure all your network connections. This includes wired, wireless, and all manual connections. The network-admin tool has been dropped. Network Interface Connection (NIC cards) hardware is detected using HAL. Information provided by Network Manager is made available to other applications over D-Bus.

Network Manager will display a Network applet icon to the right on the top panel. Left-click to see a list of all possible network connections; including all available wireless connections (see Figure 28-1). Password-protected access points will display a lock next to them. The VPN Connection entry submenu will list configured VPN connection for easy access. The Configure VPN eatery will open Network Manager to the VPN tab where you can then add, edit, or delete VPN connections. The Disconnect VPN entry will end the current active VPN connection.

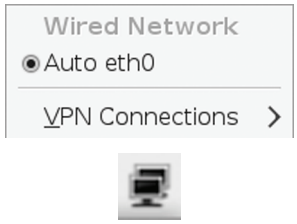


Figure 28-1: NetworkManager network connections menu and applet icon

Right-click to have the option of editing your connection, shutting off your connection (enable networking), or to see information about the connection. Select Edit connections to use Network Manager's own manual network configuration capability, which includes both wired connections and wireless 3G configuration (see Figure 28-2).

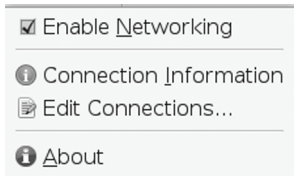


Figure 28-2: NetworkManager GNOME options

Note: The KDE version of Network Manager, knetworkmanager, also detects network connections. In addition it allows you to configure PPP dial up connections as well as manage wireless connections. To start kdenetworkmanager, select its entry in Applications | Network Tools.

You can also select the System | Preferences | Network Configuration menu entry.

With multiple wireless access points for Internet connections, a system could have several different network connections to choose from, instead of a single-line connection like DSL or cable. This is particularly true for notebook computers that could access different wireless connections at different locations. Instead of manually configuring a new connection each time one is encountered, the Network Manager tool can automatically configure and select a connection to use.

By default, an Ethernet connection will be preferred if available. Direct lines that support Ethernet connections are normally considered faster than wireless ones. For wireless connections, you will need to choose the one you want.

Network Manager is designed to work in the background, providing status information for your connection and switching from one configured connection to another as needed. For initial configuration, it detects as much information as possible about the new connection. It operates as a GNOME Panel applet, monitoring your connection, and can work on any Linux distribution.

Network Manager operates as a daemon with the name NetworkManager. If no Ethernet connection is available, Network Manager will scan for wireless connections, checking for Extended Service Set Identifiers (ESSIDs). If an ESSID identifies a previously used connection, then it is automatically selected. If several are found, then the most recently used one is chosen. If only a new connection is available, then Network Manager waits for the user to choose one. A connection is selected only if the user is logged in. If an Ethernet connection is later made, then Network Manager will switch to it from wireless.

Network Manager operates as a daemon with the name NetworkManager. It is managed with the NetworkManager service script, which you can start and stop using the Services tool in System | Administration | Services, or by using the service command in a terminal window.

```
service NetworkManager start
```

To have it start up automatically, you can use the Services tool, System | Administration | Services, or use **chkconfig**.

```
chkconfig NetworkManager on
```

Network Interface Connection (NIC cards) hardware is detected using HAL. Information provided by Network Manager is made available to other applications over D-Bus. Features currently under development include VPN and application notification. Network Manager uses the DHCP6c client to gather network information. For user interaction and notification, it uses NetworkManagerInfo.

Note: The KDE version of Network Manager, knetworkmanager, also detects network connections. To start kdenetworkmanager, select its entry in Applications | Network Tools.

Network Manager manual configuration for all network connections

Should you need to edit any network connection, wired or wireless, you right-click on the Network Manager icon and select Edit Network Connections (see Figure 28-2). This opens Network Manager's Network Connections window as shown in Figure 28-3. Established connections will be listed, with Add, Edit, and Delete buttons for adding, editing, and removing network connections. Your current network connections should already be listed, having been

automatically detected by udev and HAL. In the Figure 28-3 a wired Ethernet connection referred to as **eth0** is listed, the first Ethernet connection.

Tip: There is no longer a Networking entry in the System | Administration menu. To configure your network, access Network Manager by clicking its top panel applet, the black computer icon.

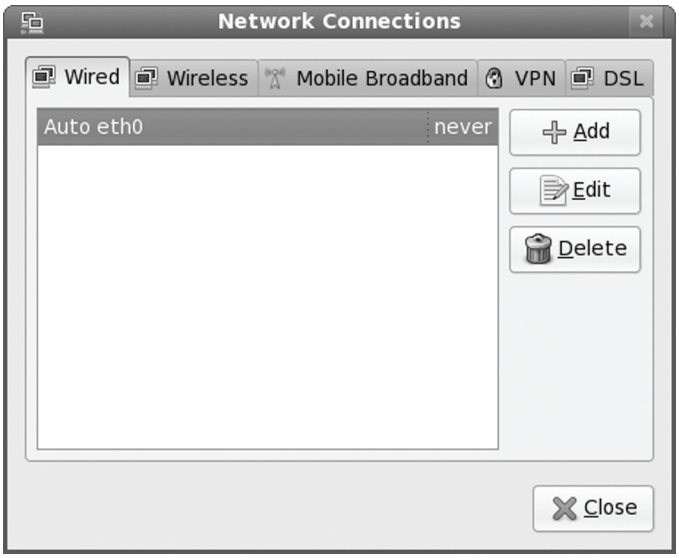


Figure 28-3: Network configuration

On the Network Connections window there are three tabs: Wired, Wireless, Mobile Broadband, VPN, and DSL.

- **Wired**: The wired connection is used for the standard IPv4 and IPv6 Ethernet connections, featuring support for DHCP and manual Ethernet settings.
- **Wireless**: The Wireless tab is where you enter in wireless configuration data like your ESSID, password, and encryption method.
- **Mobile Broadband**: The mobile broadband tab is the wireless 3G configuration, with selection for the service you are using.
- **VPN**: The VPN tab lets you specify a virtual private network
- **DSL**: The DSL tab lets you set up a direct DSL connection.

Each tab will list their configured network connections. Click on the ADD or EDIT buttons to configure a new network connection of that type, or edit an existing one.

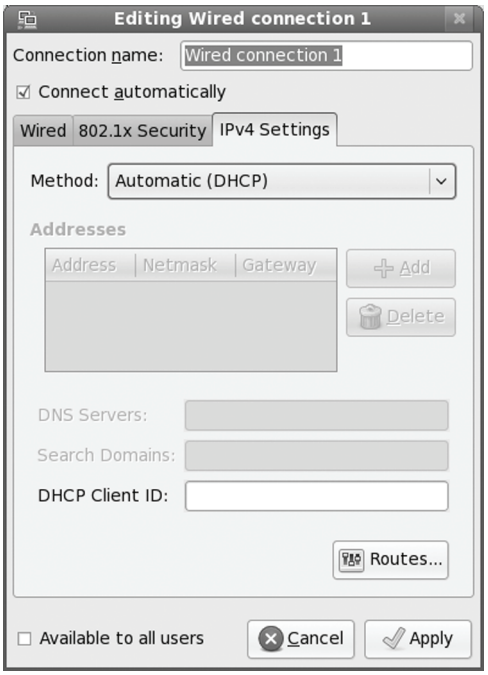


Figure 28-4: DHCP Wired Configuration

Wired Configuration

To edit a wired connection, select the connection and click the EDIT buttons on the Wired tab. This opens an Editing window as shown in Figure 28-4.

The ADD button is used to add a new connection and opens a similar window, with no settings. The Wired tab lists the MAC hardware address and the MTU. The MTU is usually set to automatic. Figure 28-4 shows the standard default configuration for a wired Ethernet connection using DHCP. Connect automatically will set up the connection when the system starts up.

There are three tabs, Wired, 8.02.1x Security, and IPv4 Settings.

The IPv4 Setting tab lets you select the kind of wired connection you have. The options are:

- **Automatic (DHCP)**: DHCP connection. Address information is blocked out.
- **Automatic (DHCP) addresses only**: DHCP that lets you specify your DNS server addresses.
- **Manual**: Enter your IP, network, and gateway addresses along with your DNS server addresses and your network domain name.
- **Link-local only**: IPv6 private local network (like IPv4 192.18.0 addresses). All address entries are blocked out.
- **Shared to other computers**: All address entries are blocked out.

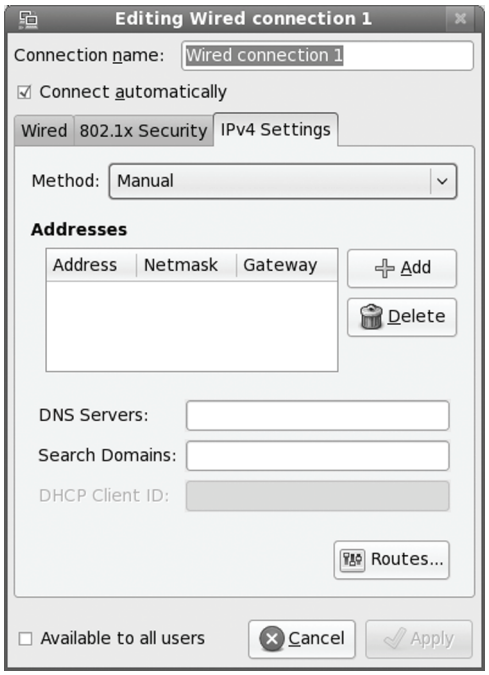


Figure 28-5: Manual Wired Configuration

Figure 28-5 shows the manual configuration entries for a wired Ethernet connection. Click the Add button to enter the IP address, network mask, and gateway address. Then enter the address for the DNS servers and your network search domains.

Figure 28-6: 802.1 Security Configuration

The Routes button will open a window where you can manually enter any network routes.

The 802.1 tab allows you to configure 802.1 security, if your network supports it (see Figure 28-6).

Wireless Configuration

For wireless connections, you click ADD or EDIT on the Network Connections window's Wireless tab. The Editing Wireless connection window opens with tabs for your wireless information, security, and IPv4 settings (See Figure 28-7). On the Wireless tab you specify your SSID, along with your Mode and MAC address.

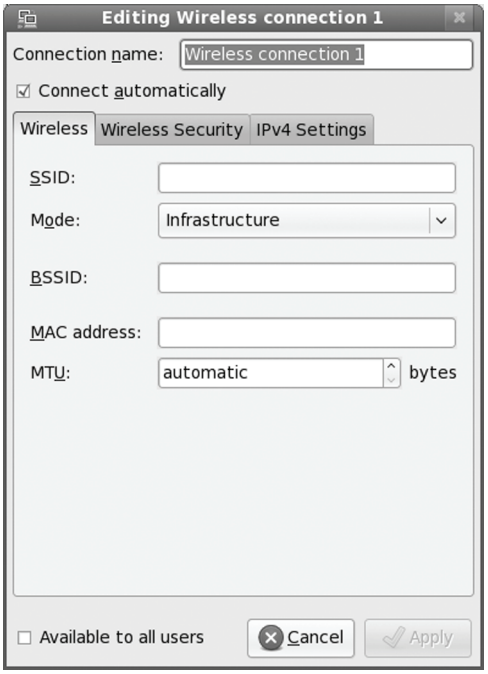


Figure 28-7: Wireless configuration

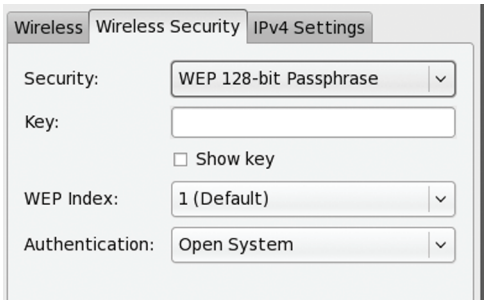


Figure 28-8: Wireless Security

On the Wireless Security tab you enter your wireless connection security method (see Figure 28-8). The commonly used method, WEP, is supported, along with WPA personal. The WPA personal method only requires a password. More secure connections like Dynamic WEP and

Enterprise WPA are also supported. These will require much more configuration information like authentication methods, certificates, and keys.

On the IPv4 Settings tab you enter your wireless connection's network address settings. This tab is the same as the IPv4 Setting on the Wired connection (see Figures 28-5 and 28-6). You have the same options: DHCP, DHCP with DNS addresses, Manual, Link-local only, and Shared.

DSL Configuration

For a direct DSL connection, you click ADD or EDIT on the Network Connections window's DSL tab. The DSL connection window will open, showing tabs for both DSL and wired PPP connections. Here you enter your DSL user name, service provider, and password (see Figure 28-9).

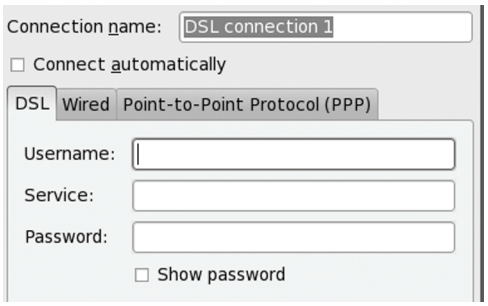


Figure 28-9: DSL manual configuration

Mobile Broadband: 3G Support

To set up a Mobile Broadband 3G connections, you select the Mobile Broadband tab in the Network Connections window, and click the ADD or EDIT button. You first choose either a CDMA or GSM 3G connection (see Figure 28-10).

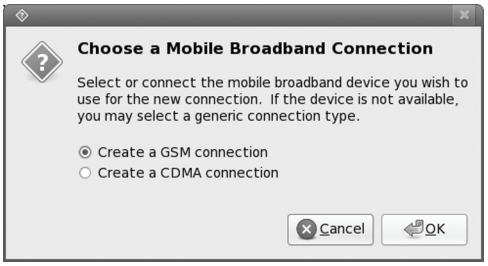


Figure 28-10: Selecting 3G Support

The connection window for each will provide panels for both Mobile Broadband and PPP. On the Mobile Broadband panel you can enter your number, user name, and password.

The GSM connection window has advanced options including the APN, Network, PIN, and PUK (see Figure 28-11). The PPP tab lets you set up your connection configuration.

Figure 28-11: 3G GSM Support

For CDMA connections, the Mobile Broadband tab provides entries for just the number, username, and password (see Figure 28-12).

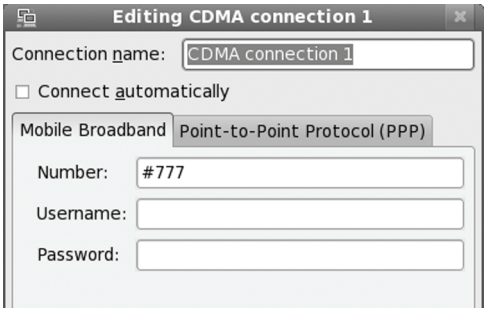


Figure 28-12: 3G CDMA Support

Once a service is selected, you can further edit the configuration by clicking its entry in the Mobile Broadband tab and clicking the EDIT button. The Editing window opens with tabs for Mobile Broadband and PPP. On the Mobile Broadband panel you can enter your number, user name, and password. Advance options including the APN, Network, type, PIN, and PUK. The APN should already be entered.

PPP Configuration

For either Wireless Broadband or DSL connections you can also specify PPP information. The PPP tab is the same for both (See Figure 28-13). There are sections Authentication, Compressions, and the Echo option. Check what features are supported by your particular PPP connection.

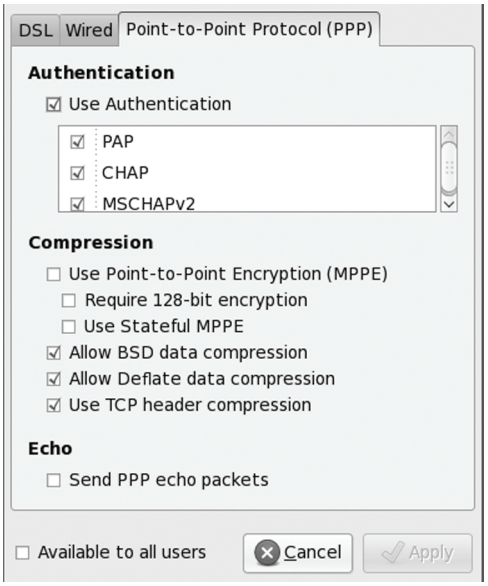


Figure 28-13: PPP Configuration

VPN Configuration

For Virtual Private Network (VPN) connections, click on the VPN tab in the Network Connections window. First make sure you have installed the **openvpn** and **NetworkManager-openvpn** packages. On the VPN tab, you can click the Add button to set up a new VPN connection, or click the Import button to import a previously configured connection. To set up a new connection, first you select the VPN type, of which OpenVPN is one. Click the Create button to open an edit window for the VPN connection. Here you can name the connection and choose to connect automatically. A VPN tab lets you select the Gateway and Authentication information, including the certificates, keys, and password. The Advanced button lets you specify certain features like a custom gateway port and a cipher for TLS certificates.

On the IPv4 tab you provide network information. You can choose Automatic VPN, Automatic which lets you provides the DHCP client id, VPN Addresses only for specifying the DNS information, and Manual where you can enter specific IP, Netmask, and Gateway addresses along with DNS addresses.

System wide network configuration: system-config-network, System | Administration | Network

Fedora 10 still provides the older system-config-network tools for easy system-wide configuration and activation of your network interfaces. You will be prompted for administrative access, and changes will be made system-wide. You can use this tool to configure and control any kind of network connection, including Ethernet cards, modems, DSL and ISDN modems, and wireless connections. All are supported with standard configuration tabs like those for IP address settings, along with specialized panels used only for a particular kind of connection, such as Compression for modem connections or Wireless Settings for a wireless card. New connections are initially configured using the Internet Configuration Wizard, which will detect and prompt for basic configuration information and then place you in the system-config-network tool to let you refine your configuration, making or changing entries as you require.

You can access the system-config-network tool directly from the System Administration menu (Network entry), System | Administration | Network. This tool opens a Network Configuration window that has five tabs: Devices, Hardware, IPsec, Hosts, and DNS (see Figure 28-14). These panels are used for configuring the network settings for your entire system. The Devices panel lists all your network connections, and Hardware lists all the network components on your system, such as Ethernet cards and modems. The DNS tab is where you enter your own system's hostname and your network's name server addresses. The Hosts tab lists static host IP addresses and their domain names, including those for your own system. The IPsec tab is used to create virtual private networks (VPNs), creating secure connections between hosts and local networks across a larger network such as the Internet. It appears only if you have installed IPsec tools and support on your system.

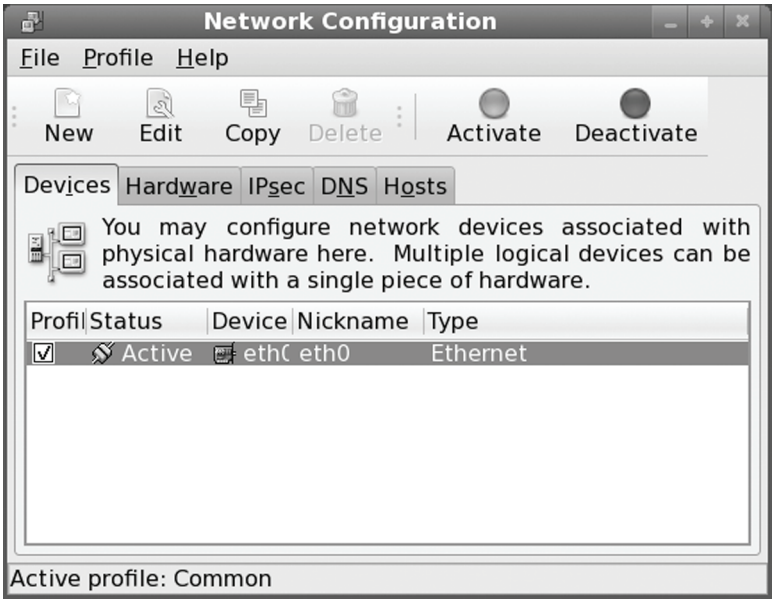


Figure 28-14: The system-config-network Network Configuration: Devices tab

DNS Settings

The DNS tab has a box at the top, labeled Hostname (see Figure 28-15). Here, you enter your system's fully qualified domain name. There are boxes for entering the IP addresses for your system's primary, secondary, and tertiary DNS servers, needed for static configurations. You can then list your search domain. Both the search domain and the name server addresses are saved in the **/etc/resolv.conf** file.

Hosts

You use the Hosts tab to associate static IP addresses with certain hosts. The tab has a single pane with New, Edit, Copy, and Delete buttons. This tab lists entries that associate hostnames with static IP addresses. You can also add aliases (nicknames). The Hosts tab actually displays the contents of the **/etc/hosts** file and saves any entries you make to that file. To add an entry, click New. A window opens with boxes for the hostname, IP address, and nicknames. When you finish, the entry is added to the Hosts list. To edit an entry, click Edit and a similar window opens, enabling you to change any of the fields. To delete an entry, select it and click Delete.

Note: If you are having trouble connecting with an Ethernet device using a static network connection, make sure that the Hosts tab lists your hostname and IP address, not just localhost. If your hostname is not there, add it.

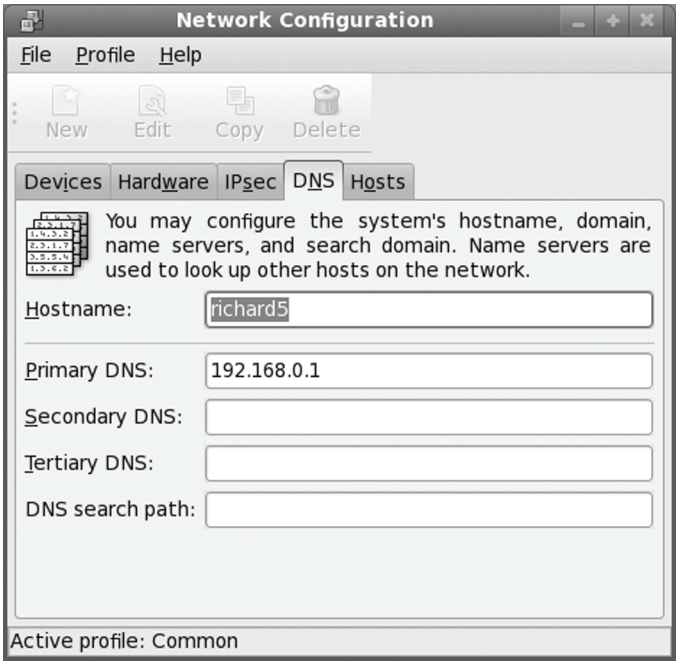


Figure 28-15: The system-config-network: DNS tab

Device Configuration: Automatic or Static

The Devices tab will list the configured network devices on your system. An entry shows the device name and its type. Use the New, Edit, Copy, and Delete buttons to manage the device

entries. To edit a device, you can just double-click its entry. For example, when you edit an Ethernet device, you click a tab to configuring it, enabling you to specify whether it is dynamic or static (see Figure 28-16). There is an entry for automatically activating it when the system starts. You can choose to use IPv6 for automatic addressing. For DHCP you can automatically obtain DNS information. For a static connection you will be able to enter an IP address, netmask, and gateway. A Hardware tab will let you choose the actual hardware device to use. The configuration tabs will differ depending on the device you edit. For example, a modem device will add tabs for provider, compression, and modem options, whereas a DSL connection will have tabs for provider, route (gateway), and hardware device. An Ethernet connection will have only general, route, and hardware device tabs. Making entries here performs the same function as **`ifconfig`**.

When you finish and are ready to save your configuration, select the Save entry from the File menu. If you want to abandon the changes you made, you can close without saving. You can run system-config-network at any time to make changes in your network configuration.

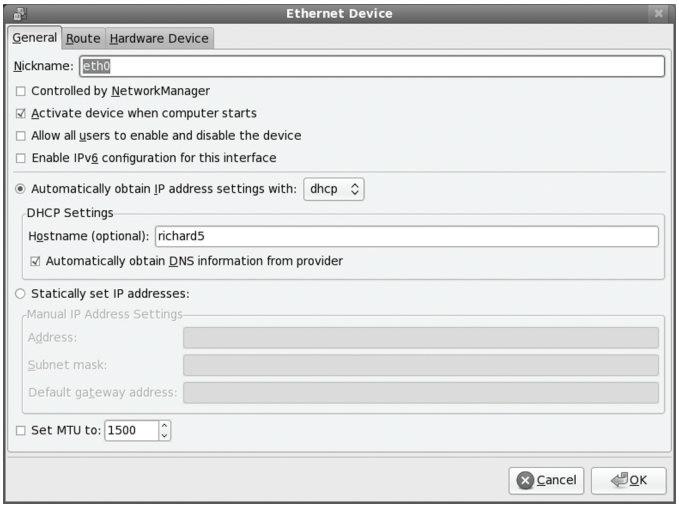


Figure 28-16: Device configuration in system-config-network

Profiles

The system-config-network tool also supports profiles. *Profiles* are commonly used for portable computers that may be moved from one environment to another. For example, at your office you could have an Office profile that uses an Ethernet card to connect to the office LAN. At home, you could use a Home profile that uses a modem to connect to the Internet. Profiles are integrated into the configuration process, with a Common profile functioning as the default configuration. The Common profile will be inherited by all other profiles, so make your basic configuration with that profile.

Profiles are accessed from the Profile menu. Select the profile you want, or select New to create a new profile. The name of the currently selected profile will be displayed at the bottom of the Network Configuration screen. The Delete entry in the Profile menu will delete the current profile. By default, the Common profile will be selected. To create a profile, click the New entry

and enter the name of the profile. It will be added to the Profile menu. You can also remove or rename a profile. The new profile will inherit the setting of the common profile. Once you have selected a profile, you can then select devices or change DNS or host information. On the Devices tab of system-config-network, each device entry will have a check box. Selecting this check box selects the device for the current profiles. To select a device for a given profile, first be sure to select the profile you are configuring, and then click the device's check box. For other profiles, the device will be unchecked. Select the Save entry from the File menu when you are finished. The changes you make will be part of that profile, and not of any other.

Configuring New Network Devices Manually

If the device was detected but not configured, you will need to configure it manually. In this case the Hardware device will appear in the Hardware tab, but not in the Devices tab. To configure a new network connection manually, first open system-config-network to the Devices tab and click New. The Add New Device Type window opens and displays a list of all possible network connections (see Figure 28-17). When you select an entry, tabs will prompt you to enter basic information about a connection; this will include phone number, username, and password for modem, DSL, and ISDN connections, whereas Ethernet connections will prompt only for IP addresses. The type of connections currently supported are Ethernet (**eth**), ISDN (**ippp**), Modem (**ppp**), xDSL (**ppp**), Token Ring (**tr**), and wireless connections (**eth**). After completing your entries, the wizard will configure your connection setting and the new connection will be ready for use.

Modem configuration

You can also use a modem with telephone lines to connect to a network. For a modem connection, the Internet Configuration Wizard will probe and detect your modem. A window will then display entries for the serial device, baud rate, hardware control, and modem volume, which you can modify. You are then prompted to enter the phone number, provider, username, and password for your ISP account. The system-config-network tool is then started up, listing your modem device as a **ppp** connection (**ppp** stands for the *Point-to-Point Protocol [PPP]* protocol that transmits IP communications across telephone lines). You can then edit the **ppp** device to modify your settings and enter any other settings; for instance, you can enter IP addresses for static connections, specify compression methods, or list your DNS servers.

DSL and ISDN Configuration

To configure DSL, you will need to provide login and password information for DSL (Digital Subscriber Line) and ISDN. In other respects, DSL and ISDN connections operate much like a local area network (LAN), treating a host as an integrated part of a network. On Fedora, you can use the Internet Configuration Wizard to set up a DSL or ISDN connection. For DSL, the Internet Configuration Wizard will display a dialog box with entries for entering your login name, your password, and the Ethernet interface your DSL modem is attached to. You will also need to enter the IP addresses for the DNS servers provided by your ISP. You can elect to have the connection automatically made up when your system starts up (depending upon your selected network profile).

Figure 28-17: Add a new network device

Wireless Configuration

Wireless connections are now detected and configured by Network Manager, both automatically and manually. Should you need to, you can also configure a wireless connection manually for system wide settings with the Internet Wizard as shown here. A wireless connection operates much like a standard Ethernet connection, requiring only an IP address and DNS server information to connect to the Internet. In addition, you will have to specify wireless connection information such as the network name and channel used. To add a new wireless connection, you start the Internet Configuration Wizard (from the System Tools menu) and select the wireless connection. If system-config-network is open, you can click Add on the Devices tab to start the Internet Configuration Wizard. You are prompted to select your wireless card.

On the Configure Wireless Connection tab, you then configure your wireless connection, selecting the mode, network name, channel, transmit, and key information.

- **Mode** Normally, you can leave this on Auto. For a simple network, one that does not require roaming, the mode is usually Ad Hoc. Managed networks allow roaming among different access points.
- **Network Name (SSID)** Can be left on Auto for a simple network. The Network Name is used to identify a cell as part of a virtual network.
- **Channel** Starting from 1, choose one with least interference.
- **Transmit Rate** Usually set to Auto to adjust automatically to degraded transmissions. But you can set a specific rate such as 11M or 1M from the pop-up menu.
- **Key** This is the encryption key for your wireless network. It must be the same for each cell on your network.

On the Configure Network Settings tab, you specify your IP address, whether it is obtained automatically with DHCP or one you enter yourself. For most company wireless networks, the IP address will be obtained automatically. Normally, the DNS servers are also provided. You can, if you wish, also specify a host name.

If you are setting up a local or home network, you will most likely use static IP addresses you select yourself from the private IP pool, beginning with 192.168, such as 192.168.0.1. The static subnet mask for a small local network is usually 255.255.255.0. The Gateway is the IP address for the computer on your network that connects to the Internet, or to a larger network.

You can later edit a wireless connection, making changes. Wireless configuration has the same General and Hardware Device tabs as an Ethernet or DSL connection, but instead of a Route tab, it has a Wireless Settings tab, where you can set your mode and network name along with channel, transmit, and key information.

Your configuration setting will be saved in an Ethernet configuration file in the **/etc/sysconfig/network-scripts** directory. For example, if your wireless card is designated **eth1**, then its configuration information is saved in the **ifcfg-eth1** file. Here you will find the standard Ethernet connection parameters such as the IP address and gateway, as well as wireless parameters such as the channel used, the mode specified, and the encryption key. The standard setting can be modified using system-config-network on that device. You could also modify this file directly to enter additional parameters, like the frequency (FREQ) or sensitivity level (SENS). You can also specify any of the **iwconfig** parameters using the IWCONFIG option. Enter **IWCONFIG** followed by and assignment of an option with a value. For example, the following option sets the fragment threshold for packets:

```
IWCONFIG="frag 512"
```

Virtual Private Networks

A virtual private network lets you create your own private logical network on top of physical network connections, such as the Internet. Using encryption, your private network transmissions are kept secure from the physical network. Though a virtual private network (VPN) has no physical connections of its own and is not an actual network, the secure transmissions it sends have the effect of operating as if the network did exist as separate entity. VPNs make use of tunneling, in which secure transmissions are sent directly through interconnecting systems on a large network like the Internet without being intercepted or, at any point, translated. To implement a VPN, each node has to use the same encryption support software. On Fedora you can choose to use either the IPsec tools. To use IPsec to set up a VPN, you click the IPsec tab in system-config-network and create a new connection.

Interface Configuration Scripts: /etc/sysconfig/network-scripts

Network configuration implemented by the Internet Configuration Wizard and system-config-network are saved in interface configuration scripts located in the **/etc/sysconfig/network-scripts** directory. You can edit these scripts directly, changing specific parameters, as discussed previously for wireless connection. Interface configuration files bear the names of the network interfaces currently configured, such as **ifcfg-eth0** for the first Ethernet device, or **ifcfg-ppp0** for the first PPP modem device. These files define shell variables that hold information on the interface, such as whether to start them at boot time. For example, the **ifcfg-eth0** file holds

definitions for NETWORK, BROADCAST, and IPADDR, which are assigned the network, broadcast, and IP addresses that the device uses. You can also manually edit these interface configuration files, making changes as noted previously for the wireless connection. A sample **ifcfg-eth0** file is shown here using a DHCP address.

/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
BOOTPROTO=DHCP
HWDADDR=00:00:00:EF:AF:00
ONBOOT=yes
TYPE=Ethernet
```

Figure 28-18: Network Device Control

system-control-network: System | Administration | Network Device Control

If you want to manually turn on or off your network connections, you use the Network Device Control tool (system-control-network), System | Administration | Network Device Control (see Figure 28-18). All your connections will be listed (many systems will have only one). You can click the Deactivate button to turn off a selected active connection, and the Activate button to turn it on. Clicking the Configure button will start up system-config-network allowing you to configure the selected device (see the following section).

Command Line PPP Access: wvdial

If, for some reason, you have been unable to set up a modem connection on your Desktop, you may have to set it up from the command line interface. For a dial-up PPP connection, you can

use the wvdial dialer, which is an intelligent dialer that not only dials up an ISP service but also performs login operations, supplying your username and password.

Variable	Description
Inherits	Explicitly inherits from the specified section. By default, sections inherit from the [Dialer Defaults] section.
Modem	The device wvdial should use as your modem. The default is **/dev/modem**.
Baud	The speed at which wvdial communicates with your modem. The default is 57,600 baud.
Init1...Init9	Specifies the initialization strings to be used by your modem; wvdial can use up to 9. The default is "ATZ" for Init1.
Phone	The phone number you want wvdial to dial.
Area Code	Specifies the area code, if any.
Dial Prefix	Specifies any needed dialing prefix—for example, 70 to disable call waiting or 9 for an outside line.
Dial Command	Specifies the dial operation. The default is "ATDT".
Login	Specifies the username you use at your ISP.
Login Prompt	If your ISP has an unusual login prompt, you can specify it here.
Password	Specifies the password you use at your ISP.
Force Address	Specifies a static IP address to use (for ISPs that provide static IP addresses to users).
Stupid Mode	In Stupid Mode, wvdial does not attempt to interpret any prompts from the terminal server and starts pppd after the modem connects.
Auto Reconnect	If enabled, wvdial attempts to reestablish a connection automatically if you are randomly disconnected by the other side. This option is on by default.

Table 28-2: Variables for wvdial

The wvdial program first loads its configuration from the **/etc/wvdial.conf** file. In here, you can place modem and account information, including modem speed and serial device, as well as ISP phone number, username, and password. The **wvdial.conf** file is organized into sections, beginning with a section label enclosed in brackets. A section holds variables for different parameters that are assigned values, such as **`username = chris`**. The default section holds default values inherited by other sections, so you needn't repeat them. Table 28-2 lists the wvdial variables.

You can use the wvdialconf utility to create a default **wvdial.conf** file for you automatically; wvdialconf will detect your modem and set default values for basic features. You can then edit the **wvdial.conf** file and modify the Phone, Username, and Password entries with your ISP dial-up information. Remove the preceding semicolon (**;**) to unquote the entry. Any line beginning with a semicolon is ignored as a comment.

```
$ wvdialconf
```

/etc/wvdial.conf

```
[Modem0]
Modem = /dev/ttyS0
Baud = 57600
Init1 = ATZ
SetVolume = 0
Dial Command = ATDT

[Dialer Defaults]
Modem = /dev/ttyS0
Baud = 57600
Init1 = ATZ
SetVolume = 0
Dial Command = ATDT

[Dialer myisp]
Username = chris
Password = mypassword
Modem = /dev/ttyS0
Phone = 555-5555
Area Code = 555
Baud = 57600
Stupid mode = 0
```

You can also create a named dialer, such as *myisp* in the following example. This is helpful if you have different ISPs you log in to. The following example shows the **/etc/wvdial.conf** file:

To start wvdial, enter the command **wvdial**, which then reads the connection configuration information from the **/etc/wvdial.conf** file; wvdial dials the ISP and initiates the PPP connection, providing your username and password when requested.

```
$ wvdial
```

You can set up connection configurations for any number of connections in the **/etc/wvdial.conf** file. To select one, enter its label as an argument to the **wvdial** command, as shown here:

```
$ wvdial myisp
```

Manual Wireless Configuration with iwconfig

NetworkManager will automatically detect and configure your wireless connections. However, you can manually configure your connections with wireless tools like system-config-network (wireless properties), Kwifimanager, and ifwconfig. Wireless configuration using system-config-network was discussed in the previous section.

Wireless configuration makes use of the same set of Wireless Extensions. The Wireless Tools package is a set of network configuration and reporting tools for wireless devices installed on a Linux system. They are currently supported and developed as part of the Linux Wireless Extension and Wireless Tools Project, an open source project maintained by Hewlett-Packard.

Wireless Tools

Wireless Tools consists of the configuration and report tools listed here:

Tool	Description
`iwconfig`	Sets the wireless configuration options basic to most wireless devices.
`iwlist`	Displays current status information of a device.
`iwspy`	Sets the list of IP addresses in a wireless network and checks the quality of their connections.
`iwpriv`	Accesses configuration options specific to a particular device.

The wireless LAN device will have an Ethernet name just like an Ethernet card. The appropriate modules will automatically be loaded, listing their aliases in the **/etc/modprobe.conf** file.

iwconfig

The **iwconfig** command works much like **ifconfig**, configuring a network connection. It is the tool used by the Internet Connection Wizard and by system-config-network to configure a wireless card. Alternatively, you can run **iwconfig** directly on a command line, specifying certain parameters. Added parameters let you set wireless-specific features such as the network name (nwid), the frequency or channel the card uses (freq or channel), and the bit rate for transmissions (rate). See the **iwconfig** Man page for a complete listing of accepted parameters. Some of the commonly used parameters are listed in Table 28-3.

For example, to set the channel used for the wireless device installed as the first Ethernet device, you would use the following, setting the channel to 2:

```
iwconfig eth0 channel 2
```

You can also use **iwconfig** to display statistics for your wireless devices, just as **ifconfig** does. Enter the **iwconfig** command with no arguments or with the name of the device. Information such as the name, frequency, sensitivity, and bit rate is listed. Check also **/proc/net/wireless** for statistics.

Instead of using **iwconfig** directly to set parameters, you can specify them in the wireless device's configuration file. The wireless device configuration file will be located in the **/etc/sysconfig/network-scripts** directory and given a name like **ifcfg-eth1**, depending on the name of the device. This file will already contain many **iwconfig** settings. Any further setting can be set by assigning **iwconfig** values to the IWCONFIG parameter as shown here.

```
IWCONFIG="rate 11M"
```

iwpriv

The **iwpriv** command works in conjunction with **iwconfig**, allowing you set options specific to a particular kind of wireless device. With **iwpriv**, you can also turn on roaming or select

the port to use. You use the *private-command* parameter to enter the device-specific options. The following example sets roaming on:

```
iwpriv eth0 roam on
```

Parameter	Description
essid	A network name
freq	The frequency of the connection
channel	The channel used
nwid or domain	The network ID or domain
mode	The operating mode used for the device, such as Ad Hoc, Managed, or Auto. Ad Hoc = one cell with no access point, Managed = network with several access points and supports roaming, Master = the node is an access point, Repeater = node forwards packets to other nodes, Secondary = backup master or repeater, Monitor = only receives packets
sens	The sensitivity, the lowest signal level at which data can be received
key or enc	The encryption key used
frag	Cut packets into smaller fragments to increase better transmission
bit or rate	Speed at which bits are transmitted. The auto option automatically falls back to lower rates for noisy channels
ap	Specify a specific access point
power	Power management for wakeup and sleep operations

Table 28-3: Commonly Used Parameters

iwspy

Your wireless device can check its connection to another wireless device it is receiving data from, reporting the quality, signal strength, and noise level of the transmissions. Your device can maintain a list of addresses for different devices it may receive data from. You use the **iwspy** tool to set or add the addresses that you want checked. You can list either IP addresses or the hardware versions. A **+** sign will add the address, instead of replacing the entire list:

```
iwspy eth0 +192.168.2.5
```

To display the quality, signal, and noise levels for your connections, you use the **iwspy** command with just the device name:

```
iwspy eth0
```

iwlist

To obtain more detailed information about your wireless device, such as all the frequencies or channels available, you use the **iwlist** tool. Using the device name with a particular parameter, you can obtain specific information about a device, including the frequency, access points, rate, power features, retry limits, and encryption keys used. You can use iwlist to

obtain information about faulty connections. The following example will list the frequencies used on the **eth0** wireless device.

```
iwlist eth0 freq
```

linux-wlan

The linux-wlan project (**www.linux-wlan.org**) has developed a separate set of wireless drivers designed for Prism-based wireless cards supporting the new 802.11 wireless standard. The linux-wlan drivers are not currently included with Fedora; you will have to download the drivers. The original source code package is available from the wlan-linux site at **www.linux-wlan.org**. The current package is linux-wlan-ng. You will have to unpack and compile the drivers.

The drivers will install WLAN devices, with device configurations placed in the **/etc/sysconfig/network-scripts** directory. For example, the configuration for the first WLAN device will be in the **ifcfg-wlan0** script. General wireless options are placed in the **/etc/wlan.conf** configuration file.

fedora

29. Proxy Servers: Squid

Proxy servers operate as an intermediary between a local network and services available on a larger one such as the Internet. Requests from local clients for web services can be handled by the proxy server, speeding transactions as well as controlling access. Proxy servers maintain current copies of commonly accessed web pages, speeding web access times by eliminating the need to access the original site constantly. They also perform security functions, protecting servers from unauthorized access.

Protocol	Description and Port
HTTP	Web pages, port 3128
FTP	FTP transfers through websites, port 3128
ICP	Internet Caching Protocol, port 3130
HTCP	Hypertext Caching Protocol, port 4827
CARP	Cache Array Routing Protocol
SNMP	Simple Network Management Protocol, port 3401
SSL	Secure Socket Layer

Table 29-1: Protocols Supported by Squid

Squid is a free, open source, proxy-caching server for web clients, designed to speed Internet access and provide security controls for web servers. It implements a proxy-caching service for web clients that caches web pages as users make requests. Copies of web pages accessed by users are kept in the Squid cache, and as requests are made, Squid checks to see if it has a current copy. If Squid does have a current copy, it returns the copy from its cache instead of querying the original site. If it does not have a current copy, it will retrieve one from the original site. Replacement algorithms periodically replace old objects in the cache. In this way, web browsers can then use the local Squid cache as a proxy HTTP server. Squid currently handles web pages supporting the HTTP, FTP, and SSL protocols (Squid cannot be used with FTP clients), each with an associated default port (see Table 29-1). It also supports ICP (Internet Cache Protocol), HTCP (Hypertext Caching Protocol) for web caching, and SNMP (Simple Network Management Protocol) for providing status information.

You can find out more about Squid at **www.squid-cache.org**. For detailed information, check the Squid FAQ and the user manual located at their website. The FAQ is also installed in your **/usr/share/doc** under the **squid** directory.

As a proxy, Squid does more that just cache web objects. It operates as an intermediary between the web browsers (clients) and the servers they access. Instead of connections being made directly to the server, a client connects to the proxy server. The proxy then relays requests to the web server. This is useful for situations where a web server is placed behind a firewall server, protecting it from outside access. The proxy is accessible on the firewall, which can then transfer requests and responses back and forth between the client and the web server. The design is often used to allow web servers to operate on protected local networks and still be accessible on the Internet. You can also use a Squid proxy to provide web access to the Internet by local hosts. Instead of using a gateway providing complete access to the Internet, local hosts can use a proxy to allow them just web access. You can also combine the two, allowing gateway access, but using the proxy server to provide more control for web access. In addition, the caching capabilities of Squid can provide local hosts with faster web access.

Technically, you could use a proxy server to simply manage traffic between a web server and the clients that want to communicate with it, without doing caching at all. Squid combines both capabilities as a proxy-caching server.

Squid also provides security capabilities that let you exercise control over hosts accessing your web server. You can deny access by certain hosts and allow access by others. Squid also supports the use of encrypted protocols such as SSL. Encrypted communications are tunneled (passed through without reading) through the Squid server directly to the web server.

Squid is supported and distributed under a GNU Public License by the National Laboratory for Applied Network Research (NLANR) at the University of California, San Diego. The work is based on the Harvest Project to create a web indexing system that includes a high-performance cache daemon called **cached**. You can obtain current source code versions and online documentation from the Squid home page at **www.squid-cache.org**. The Squid software package consists of the Squid server, several support scripts for services like LDAP and HTTP, and a cache manager script called **`cachemgr.cgi`**. The **`cachemgr.cgi`** lets you view statistics for the Squid server as it runs. You can set the Squid server to start up automatically using **services-admin** or **sysv-rc-conf**.

Configuring Client Browsers

Squid supports both standard proxy caches and transparent caches. With a standard proxy cache, users will need to configure their browsers to specifically access the Squid server. A transparent cache, on the other hand, requires no browser configuration by users. The cache is transparent, allowing access as if it were a normal website. Transparent caches are implemented by IPtables using net filtering to intercept requests and direct them to the proxy cache.

With a standard proxy cache, users need to specify their proxy server in their web browser configuration. For this they will need the IP address of the host running the Squid proxy server as well as the port it is using. Proxies usually make use of port 3128. To configure use of a proxy server running on the private network, you enter the following. The proxy server is running on **turtle.mytrek.com** (192.168.0.1) and using port 3128.

```
192.168.0.1 3128
```

On Firefox, Mozilla, and Netscape, the user on the sample local network first selects the Proxy tab located in Preferences under the Edit menu. Then, in the Manual proxy configuration's View tab, you enter the previous information. The user will see entries for FTP, HTTP, and security proxies. For standard web access, enter the IP address in the FTP and web boxes. For their port boxes, enter **3128**.

For GNOME, select Network Proxy in Preferences menu or window, and for Konqueror on the KDE Desktop, select the Proxies tab on the Preferences | Web Browsing menu window. Here, you can enter the proxy server address and port numbers. If your local host is using Internet Explorer (as a Windows system does), you set the proxy entries in the Local Area Network settings accessible from the Internet Options window.

On Linux or Unix systems, local hosts can set the **`http_proxy`** and **`ftp_proxy`** shell variables to configure access by Linux-supported web browsers such as Lynx. You can place these definitions in your **.bash_profile** or **/etc/profile** file to have them automatically defined whenever you log in.

```
http_proxy=192.168.0.1:3128
ftp proxy=192.168.0.1:3128
export http_proxy ftp_proxy
```

Alternatively, you can use the proxy's URL.

```
http_proxy=http://turtle.mytrek.com:3128
```

For the Elinks browser, you can specify a proxy in its configuration file, **/etc/elinks.conf**. Set both FTP and web proxy host options, as in:

```
protocol.http.proxy.host  turtle.mytrek.com:3128
protocol.ftp.proxy.host  turtle.mytrek.com:3128
```

Before a client on a local host can use the proxy server, access permission has to be given to it in the server's **squid.conf** file, described in the later section "Security." Access can easily be provided to an entire network. For the sample network used here, you would have to place the following entries in the **squid.conf** file. These are explained in detail in the following sections.

```
acl mylan src 192.168.0.0/255.255.255.0
http_access allow mylan
```

Tip: Web clients that need to access your Squid server as a standard proxy cache will need to know the server's address and the port for Squid's HTTP services, by default 3128.

The squid.conf File

The Squid configuration file is **squid.conf**, located in the **/etc/squid** directory. All options are listed with detailed explanations. Inactive options are commented out. In the **/etc/squid/squid.conf** file, you set general options such as ports used, security options controlling access to the server, and cache options for configuring caching operations. You can use a backup version called **/etc/squid/squid.conf.default** to restore your original defaults. The default version of **squid.conf** provided with Squid software includes detailed explanations of all standard entries, along with commented default entries. Entries consist of tags that specify different attributes. For example, **maximum_object_size** and **maximum_object** set limits on objects transferred.

```
maximum_object_size 4096 KB
```

As a proxy, Squid will use certain ports for specific services, such as port 3128 for HTTP services like web browsers. Default port numbers are already set for Squid. Should you need to use other ports, you can set them in the **/etc/squid/squid.conf** file. The following entry shows how you set the web browser port:

```
http_port 3128
```

Note: Squid uses the Simple Network Management Protocol (SNMP) to provide status information and statistics to SNMP agents managing your network. You can control SNMP with the **snmp access** and **port** configurations in the **squid.conf** file.

Options	Description
src *ip-address/netmask*	Client's IP address
src *addr1-addr2/netmask*	Range of addresses
dst *ip-address/netmask*	Destination IP address
myip *ip-address/netmask*	Local socket IP address
srcdomain *domain*	Reverse lookup, client IP
dstdomain *domain*	Destination server from URL; for **dstdomain** and **dstdom_regex**, a reverse lookup is tried if an IP-based URL is used
srcdom_regex [-i] *expression*	Regular expression matching client name
dstdom_regex [-i] *expression*	Regular expression matching destination
time *[day-abbrevs] [h1:m1-h2:m2]*	Time as specified by day, hour, and minutes. Day abbreviations: S = Sunday, M = Monday, T = Tuesday, W = Wednesday, H = Thursday, F = Friday, A = Saturday
url_regex [-i] *expression*	Regular expression matching on whole URL
urlpath_regex [-i] *expression*	Regular expression matching on URL path
port *ports*	A specific port or range of ports
proto *protocol*	A specific protocol, such as HTTP or FTP
method *method*	Specific methods, such as GET and POST
browser [-i] *regexp*	Pattern match on user-agent header
ident *username*	String match on **ident** output
src_as *number*	Used for routing of requests to specific caches
dst_as *number*	Used for routing of requests to specific caches
proxy_auth *username*	List of valid usernames
snmp_community *string*	A community string to limit access to your SNMP agent

Table 29-2: Squid ACL Options

Proxy Security

Squid can use its role as an intermediary between web clients and a web server to implement access controls, determining who can access the web server and how. Squid does this by checking access control lists (ACLs) of hosts and domains that have had controls placed on them. When it finds a web client from one of those hosts attempting to connect to the web server, it executes the control. Squid supports a number of controls with which it can deny or allow access to the web server by the remote host's web client (see Table 29-2). In effect, Squid sets up a firewall just for the web server.

The first step in configuring Squid security is to create ACLs. These are lists of hosts and domains for which you want to set up controls. You define ACLs using the **acl** command, creating a label for the systems on which you are setting controls. You then use commands such as **http_access** to define these controls. You can define a system, or a group of systems, by use of several **acl** options, such as the source IP address, the domain name, or even the time and date. For example, the **src** option is used to define a system or group of systems with a certain source address. To define a **mylan acl** entry for systems in a local network with the addresses 192.168.0.0 through 192.168.0.255, use the following ACL definition:

```
acl mylan src 192.168.0.0/255.255.255.0
```

Once it is defined, you can use an ACL definition in a Squid option to specify a control you want to place on those systems. For example, to allow access by the mylan group of local systems to the web through the proxy, use an **http_access** option with the **allow** action specifying **mylan** as the **acl** definition to use, as shown here:

```
http_access allow mylan
```

By defining ACLs and using them in Squid options, you can tailor your website with the kind of security you want. The following example allows access to the web through the proxy by only the **mylan** group of local systems, denying access to all others. Two **acl** entries are set up: one for the local system and one for all others; **http_access** options first allow access to the local system and then deny access to all others.

```
acl mylan src 192.168.0.0/255.255.255.0
acl all src 0.0.0.0/0.0.0.0
http_access allow mylan
http_access deny all
```

The default entries that you will find in your **squid.conf** file, along with an entry for the **mylan** sample network, are shown here. You will find these entries in the ACCESS CONTROLS section of the **squid.conf** file.

```
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl mylan src 192.168.0.0/255.255.255.0
acl SSL_ports port 443 563
```

The order of the **http_access** options is important. Squid starts from the first and works its way down, stopping at the first **http_access** option with an ACL entry that matches. In the preceding example, local systems that match the first **http_access** command are allowed, whereas others fall through to the second **http_access** command and are denied.

For systems using the proxy, you can also control what sites they can access. For a destination address, you create an **acl** entry with the **dst** qualifier. The **dst** qualifier takes as its argument the site address. Then you can create an **http_access** option to control access to that address. The following example denies access by anyone using the proxy to the destination site **rabbit.mytrek.com**. If you have a local network accessing the web through the proxy, you can use such commands to restrict access to certain sites.

```
acl myrabbit dst rabbit.mytrek.com
http_access deny myrabbit
```

The **http_access** entries already defined in the **squid.conf** file, along with an entry for the **mylan** network, are shown here. Access to outside users is denied, whereas access by hosts on the local network and the local host (Squid server host) is allowed.

```
http_access allow localhost
http_access allow mylan
http_access deny all
```

Proxy Caches

Squid primarily uses the Internet Cache Protocol (ICP) to communicate with other web caches. It also provides support for the more experimental Hypertext Cache Protocol (HTCP) and the Cache Array Routing Protocol (CARP).

Using the ICP protocols, your Squid cache can connect to other Squid caches or other cache servers, such as Microsoft proxy server, Netscape proxy server, and Novell BorderManager. This way, if your network's Squid cache does not have a copy of a requested web page, it can contact another cache to see if it is there instead of accessing the original site. You can configure Squid to connect to other Squid caches by connecting it to a cache hierarchy. Squid supports a hierarchy of caches denoted by the terms *child, sibling,* and *parent.* Sibling and child caches are accessible on the same level and are automatically queried whenever a request cannot be located in your own Squid's cache. If these queries fail, a parent cache is queried, which then searches its own child and sibling caches—or its own parent cache, if needed—and so on.

You can set up a cache hierarchy to connect to the main NLANR server by registering your cache using the following entries in your **squid.conf** file:

```
cache_announce 24
announce_to sd.cache.nlanr.net:3131
```

Use **cache_peer** to set up parent, sibling, and child connections to other caches. This option has five fields. The first two consist of the hostname or IP address of the queried cache and the cache type (parent, child, or sibling). The third and fourth are the HTTP and the ICP ports of that cache, usually 3128 and 3130. The last is used for cache_peer options such as proxy-only to not save fetched objects locally, no-query for those caches that do not support ICP, and weight, which assigns priority to a parent cache. The following example sets up a connection to a parent cache:

```
cache_peer sd.cache.nlanr.net parent 3128 3130
```

Squid provides several options for configuring cache memory. The **cache_mem** option sets the memory allocated primarily for objects currently in use (objects in transit). If available, the space can also be used for frequently accessed objects (hot objects) and failed requests (negative-cache objects). The default is 8MB. The following example sets it to 256MB:

```
cache_mem 256 MB
```

Logs

Squid keeps several logs detailing access, cache performance, and error messages.

- **access.log** holds requests sent to your proxy.
- **cache.log** holds Squid server messages such as errors and startup messages.
- **store.log** holds information about the Squid cache such as objects added or removed.

You can use the cache manager (**cachemgr.cgi**) to manage the cache and view statistics on the cache manager as it runs. To run the cache manager, use your browser to execute the **cachemgr.cgi** script (this script should be placed in your web server's **cgi-bin** directory).

fedora

30. Domain Name System

The Domain Name System (DNS) is an Internet service that locates and translates domain names into their corresponding Internet Protocol (IP) addresses. As you may recall, all computers connected to the Internet are addressed using an IP address. As a normal user on a network might have to access many different hosts, keeping track of the IP addresses needed quickly became a problem. It was much easier to label hosts with names and use the names to access them. Names were associated with IP addresses. When a user used a name to access a host, the corresponding IP address was looked up first and then used to provide access.

With the changeover from IPv4 to IPv6 address, DNS servers will have some configuration differences. Both are covered here, though some topics will use IPv4 addressing for better clarity, as they are easier to represent.

DNS Address Translations

The process of translating IP addresses into associated names is fairly straightforward. Small networks can be set up easily, with just the basic configuration. The task becomes much more complex when you deal with larger networks and with the Internet. The sheer size of the task can make DNS configuration a complex operation.

Fully Qualified Domain Names

IP addresses were associated with corresponding names, called fully qualified domain names. A *fully qualified domain name* is composed of three or more segments. The first segment is the name that identifies the host, and the remaining segments are for the network in which the host is located. The network segments of a fully qualified domain name are usually referred to simply as the domain name, while the host part is referred to as the hostname (though this is also used to refer to the complete fully qualified domain name). In effect, subnets are referred to as domains. The fully qualified domain name **www.linux.org** could have an IPv4 address 198.182.196.56, where 198.182.196 is the network address and 56 is the host ID. Computers can be accessed only with an IP address, so a fully qualified domain name must first be translated into its corresponding IP address to be of any use. The parts of the IP address that make up the domain name and the hosts can vary.

IPv4 Addressing

The IP address may be implemented in either the newer IPv6 (Internet Protocol Version 6) format or the older and more common IPv4 (Internet Protocol Version 4) format. Since the IPv4 addressing is much easier to read, that format will be used in these examples. In the older IPv4 format, the IP address consists of a number composed of four segments separated by periods. Depending on the type of network, several of the first segments are used for the network address and one or more of the last segments are used for the host address. In a standard class C network used in smaller networks, the first three segments are the computer's network address and the last segment is the computer's host ID (as used in these examples). For example, in the address 192.168.0.2, 192.168.0 is the network address and 2 is the computer's host ID within that network. Together, they make up an IP address by which the computer can be addressed from anywhere on the Internet. IP addresses, though, are difficult to remember and easy to get wrong.

IPv6 Addressing

IPv6 addressing uses a very different approach designed to provide more flexibility and support for very large address spaces. There are three different types of IPv6 addresses, unicast, multicast, and anycast, of which unicast is the most commonly used. A unicast address is directed to a particular interface. There are several kinds of unicast addresses, depending on how the address is used. For example, you can have a global unicast address for access through the Internet or a site-level unicast address for private networks.

Though consisting of 128 bits in eight segments (16 bits, 2 bytes, per segment), an IPv6 address is made up of several fields that conform roughly to the segments and capabilities of an IPv4 address, networking information, subnet information, and the interface identifier (host ID). The network information includes a format prefix indicating the type of network connection. In addition, a subnet identifier can be used to specify a local subnet. The network information takes up the first several segments; the remainder is used for the interface ID. The interface ID is a 64-bit (four-segment) Extended Unique Identifier (EUI-64) generated from a network device's Media Access Control (MAC) address. IP addresses are written in hexadecimal numbers, making them difficult to use. Each segment is separated from the next by a colon, and a set of consecutive segments with zero values like the reserved segment can be left empty.

Manual Translations: /etc/hosts

Any computer on the Internet can maintain a file that manually associates IP addresses with domain names. On Linux and UNIX systems, this file is called the **/etc/hosts** file. Here, you can enter the IP addresses and domain names of computers you commonly access. Using this method, however, each computer needs a complete listing of all other computers on the Internet, and that listing must be updated constantly. Early on, this became clearly impractical for the Internet, though it is still feasible for small, isolated networks as well as simple home networks.

DNS Servers

The Domain Name System has been implemented to deal with the task of translating the domain name of any computer on the Internet to its IP address. The task is carried out by interconnecting servers that manage the Domain Name System (also referred to as DNS servers or name servers). These DNS servers keep lists of fully qualified domain names and their IP addresses, matching one up with the other. This service that they provide to a network is referred to as the Domain Name System. The Internet is composed of many connected subnets called *domains,* each with its own Domain Name System (DNS) servers that keep track of all the fully qualified domain names and IP addresses for all the computers on its network. DNS servers are hierarchically linked to root servers, which, in turn, connect to other root servers and the DNS servers on their subnets throughout the Internet. The section of a network for which a given DNS server is responsible is called a *zone.* Although a zone may correspond to a domain, many zones may, in fact, be within a domain, each with its own name server. This is true for large domains where too many systems exist for one name server to manage.

DNS Operation

When a user enters a fully qualified domain name to access a remote host, a resolver program queries the local network's DNS server requesting the corresponding IP address for that remote host. With the IP address, the user can then access the remote host. In Figure 30-1, the user

at **rabbit.mytrek.com** wants to connect to the remote host **lizard.mytrek.com**. **rabbit.mytrek.com** first sends a request to the network's DNS server—in this case, **turtle.mytrek.com**—to look up the name **lizard.mytrek.com** and find its IP address. **turtle.mytrek.com** then returns the IP address for **lizard.mytrek.com**, 192.168.0.3, to the requesting host, **rabbit.mytrek.com**. With the IP address, the user at **rabbit.mytrek.com** can then connect to **lizard.mytrek.com**.

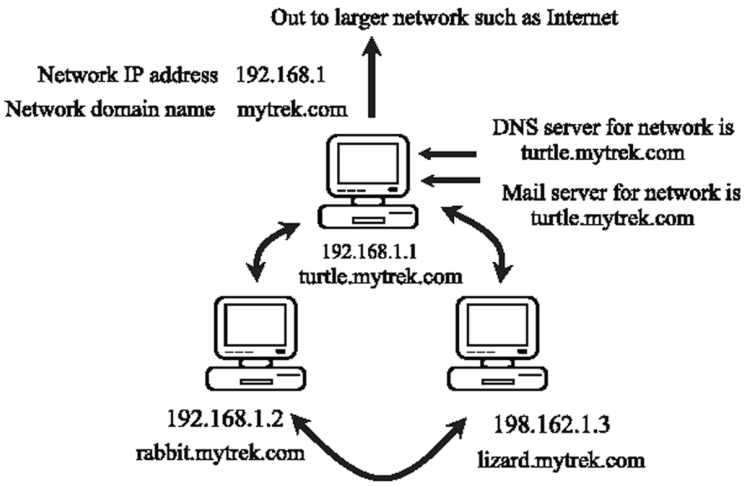


Figure 30-1: DNS server operation

DNS Clients: Resolvers

The names of the DNS servers that service a host's network are kept in the host's **/etc/resolv.conf** file. When setting up an Internet connection, the name servers provided by your Internet service provider (ISP) were placed in this file. These name servers resolve any fully qualified domain names that you use when you access different Internet sites. For example, when you enter a Web site name in your browser, the name is looked up by the name servers and the name's associated IP address is then used to access the site.

/etc/resolv.conf

```
search  mytrek.com   mytrain.com
nameserver  192.168.0.1
nameserver  192.168.0.3
```

Local Area Network Addressing

If you are setting up a DNS server for a local area network (LAN) that is not connected to the Internet, you should use a special set of IP numbers reserved for such local networks (also

known as *private networks* or *intranets*). This is especially true if you are implementing IP masquerading, where only a gateway machine has an Internet address, and the others make use of that one address to connect to the Internet. Though structurally the same, IPv4 and IPv6 use different addressing formats for local addresses. Many local and home networks still use the IPv4 format, and this is the format used in the following local addressing example.

Address	Networks
10.0.0.0	Class A network
172.16.0.0–172.31.255.255	Class B network
192.168.0.0	Class C network
127.0.0.0	Loopback network (for system self-communication)

Table 30-1: Non-Internet Private Network IP Addresses

IPv4 Private Networks

IPv4 provides a range of private addresses for the three classes supported by IPv4. As you have seen, class C IPv4 network numbers have the special network number 192.168. Numbers are also reserved for class A and class B non-Internet local networks. Table 30-1 lists these addresses. The possible addresses available span from 0 to 255 in the host segment of the address. For example, class B network addresses range from 172.16.0.0 to 172.16.255.255, giving you a total of 65,534 possible hosts. The class C network ranges from 192.168.0.0 to 192.168.255.255, giving you 254 possible subnetworks, each with 254 possible hosts. The number 127.0.0.0 is reserved for a system's loopback interface, which allows it to communicate with itself, as it enables users on the same system to send messages to each other.

These numbers were originally designed for class-based addressing. However, they can just as easily be used for Classless Interdomain Routing (CIDR) addressing, where you can create subnetworks with a smaller number of hosts. For example, the 254 hosts addressed in a class C network could be split into two subnetworks, each with 125 hosts.

IPv6 Private Networks

IPv6 supports private networks with unique-local addresses that provide the same functionality of IPv4 private addresses. The unique-local addresses have no public routing information. They cannot access the Internet. They are restricted to the site they are used on. The unique-local addresses use only three fields: a format prefix, subnet identifier, and interface identifier. A site-level address has the format prefix **fc00**. If you have no subnets, it will be set to 0. This will give you a network prefix of **fc00:0:0:0**. You can drop the set of empty zeros to give you **fc00::**. The interface ID field will hold the interface identification information, similar to the host ID information in IPv4.

```
fc00::           IPv6 unique-local prefix
```

The loopback device will have special address of **::1**, also known as localhost.

```
::1              IPv6 loopback network
```

Rather than using a special set of reserved addresses as IPv4 does, with IPv6 you only use the unique-local prefix, **fc00**, and the special loopback address, **::1**.

Tip: Once your network is set up, you can use ping6 or ping to see if it is working. The ping6 tool is designed for IPv6 addresses, whereas ping is used for IPv4.

Local Network Address Example Using IPv4

If you are setting up a LAN, such as a small business or home network, you are free to use class C IPv4 network (254 hosts or less), that have the special network number 192.168, as used in these examples. These are numbers for your local machines. You can set up a private network, such as an intranet, using network cards such as Ethernet cards and Ethernet hubs, and then configure your machines with IP addresses starting from 192.168.0.1. The host segment can range from 1 to 254, where 255 is used for the broadcast address. If you have three machines on your home network, you can give them the addresses 192.168.0.1, 192.168.0.2, and 192.168.0.3. You can then set up domain name system services for your network by running a DNS server on one of the machines. This machine becomes your network's DNS server. You can then give your machines fully qualified domain names and configure your DNS server to translate the names to their corresponding IP addresses. As shown in Figure 30-1, for example, you could give the machine 192.168.0.1 the name **turtle.mytrek.com** and the machine 192.168.0.2 the name **rabbit.mytrek.com**. You can also implement Internet services on your network such as FTP, Web, and mail services by setting up servers for them on your machines. You can then configure your DNS server to let users access those services using fully qualified domain names. For example, for the **mytrek.com** network, the Web server could be accessed using the name **www.mytrek.com**. Instead of a Domain Name System, you could have the **/etc/hosts** files in each machine contain the entire list of IP addresses and domain names for all the machines in your network. But for any changes, you would have to update each machine's **/etc/hosts** file.

BIND

The DNS server software currently in use on Linux systems is Berkeley Internet Name Domain (BIND). BIND was originally developed at the University of California, Berkeley, and is currently maintained and supported by the Internet Software Consortium (ISC). You can obtain BIND information and current software releases from its Web site at **www.isc.org**. Web page documentation and manuals are included with the software package. At the site you can also access the BIND Administration Manual for detailed configuration information. RPM packages are available at distribution FTP sites. The BIND directory in **/usr/share/doc** contains extensive documentation, including Web page manuals and examples. The Linux HOW-TO for the Domain Name System, DNS-HOWTO, provides detailed examples. Documentation, news, and DNS tools can be obtained from the DNS Resource Directory (DNSRD) at **www.dns.net/dnsrd**. The site includes extensive links and online documentation, including the *BIND Operations Guide (BOG).*

BIND 9 Administrator Reference Manual is located in the **/usr/share/doc/BIND** directory within the **arm** subdirectory in Web page format. Also check the **www.isc.org** site for the latest version.

```
/user/share/doc/bind-9.5.0/arm/Bv9ARM.html
```

Alternative DNS Servers

Several alternative DNS servers are now available. These include djbdns, noted for its security features, CustomDNS, a dynamic server implemented in Java (**http://customdns.sourceforge.net**), and Yaku-NS, an embedded server. The djbdns server (**http://cr.yp.to/djbdns.html**), written by D.J. Bernstein, is designed specifically with security in mind, providing a set of small server daemons, each performing specialized tasks. In particular, djbdns separates the name server, caching server, and zone transfer tasks into separate programs: tinydns (**www.tinydns.org**) implements the authoritative name server for a network, whereas dnscache implements a caching server that will resolve requests from DNS clients such as Web browsers. In effect, dnscache operates as the name server that your applications will use to resolve addresses. dnscache will then query tinydns to resolve addresses on your local network. Zone transfers are handled separately by **axfrdns** and **asfget**.

BIND Servers and Tools

The BIND DNS server software consists of a name server daemon, several sample configuration files, and resolver libraries. As of 1998, a new version of BIND, beginning with the series number 8.*x,* implemented a new configuration file using a new syntax. Version 9.0 adds new security features and support for IPv6. Older versions, which begin with the number 4.*x,* use a different configuration file with an older syntax. Most distributions currently install the newer 9.*x* version of BIND.

The name of the BIND name server daemon is **named**. To operate your machine as a name server, simply run the **named** daemon with the appropriate configuration. The **named** daemon listens for resolution requests and provides the correct IP address for the requested hostname. You can use the Remote Name Daemon Controller utility, **rndc**, provided with BIND to start, stop, restart, and check the status of the server as you test its configuration. **rndc** with the **stop** command stops **named** and, with the **start** command, starts it again, reading your **named.conf** file. **rndc** with the **help** command provides a list of all **rndc** commands. Configuration is set in the **/etc/rndc.conf** file. Once your name server is running, you can test it using the **dig** or **nslookup** utility, which queries a name server, providing information about hosts and domains. If you start **dig** with no arguments, it enters an interactive mode where you can issue different **dig** commands to refine your queries.

To check the syntax of your DNS server configuration and zone files, BIND provides the **named-checkconfig** and **named-checkzone** tools: **named-checkconfig** will check the syntax of DNS configuration file, **named.conf**, and **named-checkzone** will check a zone file's syntax. Other syntax checking tools are also available, such as **nslint**, which operates like the programming tool **lint**.

Numerous other DNS tools are also available. Check the DNS Resource Directory at **www.dns.net/dnsrd** for a listing. Table 30-2 lists several DNS administrative tools.

On most distributions, **named** runs as a stand-alone daemon, starting up when the system boots and constantly running. If you don't want **named** to start up automatically, you can use **chkconfig** to change its status.

Tool	Description
dig *domain*	Domain Information Groper, tool to obtain information on a DNS server. Preferred over **nslookup**
host *hostname*	Simple lookup of hosts
nslookup *domain*	Tool to query DNS servers for information about domains and hosts
named-checkconf	BIND tool to check the syntax of your DNS configuration file, **/etc/named.conf**
named-checkzone	BIND tool to check the syntax of your DNS zone files
nslint	Tool to check the syntax of your DNS configuration and zone files
rndc *command*	Remote Name Daemon Controller, an administrative tool for managing a DNS server (version 9.*x*)
ndc	Name Daemon Controller (version 8.*x*)
system-config-bind	Fedora Bind DNS server configuration tool

Table 30-2: BIND Diagnostic and Administrative Tools

Stopping and Starting the BIND server

The **named** daemon is started using a startup script in the **/etc/rc.d/init.d** directory called **named**. You can use this script to start, stop, and restart the daemon using the **stop**, **start**, and **restart** arguments. You can invoke the script with the **service** command as shown here:

```
service named restart
```

You can also reference the httpd start up script directly.

```
/etc/init.d/httpd
```

On Fedora, **named** runs as a stand-alone daemon, starting up when the system boots and constantly running. If you don't want **named** to start up automatically, you can use system-config-services (System | Administration | Services) or **chkconfig** to change its status.

Domain Name System Configuration

You configure a DNS server using a configuration file, several zone files, and a cache file. The part of a network for which the name server is responsible is called a *zone.* A zone is not the same as a domain, because in a large domain you could have several zones, each with its own name server. You could also have one name server service several zones. In this case, each zone has its own zone file.

DNS Zones

The zone files hold resource records that provide hostname and IP address associations for computers on the network for which the DNS server is responsible. Zone files exist for the server's network and the local machine. Zone entries are defined in the **named.conf** file. Here, you place zone entries for your master, slave, and forward DNS servers. The most commonly used zone types are described here:

- **Master zone** This is the primary zone file for the network supported by the DNS server. It holds the mappings from domain names to IP addresses for all the hosts on that network.
- **Slave zone** These are references to other DNS servers for your network. Your network can have a master DNS server and several slave DNS servers to help carry the workload. A slave DNS server automatically copies its configuration files, including all zone files, from the master DNS server. Any changes to the master configuration files trigger an automatic download of these files to the slave servers. In effect, you only have to manage the configuration files for the master DNS server, as they are automatically copied to the slave servers.
- **Forward zone** The forward zone lists name servers outside your network that should be searched if your network's name server fails to resolve an address.
- **in-addr.arpa zone** DNS can also provide reverse resolutions, where an IP address is used to determine the associated domain name address. Such lookups are provided by **in-addr.arpa** zone files. Each master zone file usually has a corresponding **in-addr.arpa** zone file to provide reverse resolution for that zone. For each master zone entry, a corresponding reverse mapping zone entry named **in-addr.arpa** also exists, as well as one for the localhost. This entry performs reverse mapping from an IP address to its domain name. The name of the zone entry uses the domain IP address, which is the IP address with segments listed starting from the host, instead of the network. So for the IP address 192.168.0.4, where 4 is the host address, the corresponding domain IP address is 4.0.168.192, listing the segments in reverse order. The reverse mapping for the localhost is 1.0.0.127.
- **IP6.ARPA zone** This is the IPv6 equivalent of the **in-addr.arpa** zone, providing reverse resolution for that zone. The IP6.ARPA zone uses bit labels provide a bit-level format that is easier to write, requiring no reverse calculation on the part of the DNS administrator.
- **IP6.INT zone** This is the older form of the IPv6 IP6.ARPA zone, which is the equivalent of the IPv4 **in-addr.arpa** zone, providing reverse resolution for a zone. IP6.INT is meant to be used with the older AAAA IPv6 address records. IP6.INT uses a nibble format to specify a reverse zone. In this format, a hexadecimal IPv6 address is segmented into each of its 32 hexadecimal numbers and listed in reverse order, each segment separated by a period.
- **Hint zone** A hint zone specifies the root name servers and is denoted by a period (.). A DNS server is normally connected to a larger network, such as the Internet, which has its own DNS servers. DNS servers are connected this way hierarchically, with each server having its root servers to which it can send resolution queries. The root servers are designated in the hint zone.

DNS Servers Types

There are several kinds of DNS servers, each designed to perform a different type of task under the Domain Name System. The basic kind of DNS server is the *master* server. Each network must have at least one master server that is responsible for resolving names on the network. Large networks may need several DNS servers. Some of these can be slave servers that can be updated directly from a master server. Others may be *alternative master* servers that hosts in a network can use. Both are commonly referred to as *secondary* servers. For DNS requests a DNS server cannot

resolve, the request can be forwarded to specific DNS servers outside the network, such as on the Internet. DNS servers in a network can be set up to perform this task and are referred to as *forwarder* servers. To help bear the workload, local DNS servers can be set up within a network that operate as caching servers. Such a server merely collects DNS lookups from previous requests it sent to the main DNS server. Any repeated requests can then be answered by the caching server.

A server that can answer DNS queries for a given zone with authority is known as an *authoritative* server. An authoritative server holds the DNS configuration records for hosts in a zone that will associate each host's DNS name with an IP address. For example, a master server is an authoritative server. So are slave and stealth servers (see the list that follows). A caching server is not authoritative. It only holds whatever associations it picked up from other servers and cannot guarantee that the associations are valid.

- **Master server** This is the primary DNS server for a zone.
- **Slave server** A DNS server that receives zone information from the master server.
- **Forwarder server** A server that forwards unresolved DNS requests to outside DNS servers. Can be used to keep other servers on a local network hidden from the Internet.
- **Caching only server** Caches DNS information it receives from DNS servers and uses it to resolve local requests.
- **Stealth server** A DNS server for a zone not listed as a name server by the master DNS server.

Location of Bind Server Files: /etc/named/chroot

Both the configuration and zone files used by BIND are placed in a special subdirectory called **chroot** located within the **/var/named** directory, **/var/named/chroot**. The **chroot** directory sets up a chroot jail, creating a virtual root directory for any users of the DNS service. This prevents access by DNS users to any other part of the system. When the BIND server starts up, the **chroot** command is run on the **named** service making **/var/named/chroot** the root directory for any users of the DNS service. Check the Chroot-BIND HOWTO at the Linux Documentation site for more information (**www.tldp.org**).

Within the **/var/named/chroot** directory are subdirectories that hold the BIND files on your system. These include **var**, **etc**, and **dev**. The **var** subdirectory has a **named** subdirectory within which are the zone files used by the BIND DNS server. Links to these files are located in the **/var/named** directory. The **named.conf** configuration file is located in the **/var/named/chroot/etc** directory. The **/etc/named.conf** file is just a link to this file. The links allow your configuration files and tools to reference zone files using the standard BIND DNS directory names, as in **/etc/named**. For new configuration and zone files you would create new links.

```
/var/named/chroot/etc/named.conf          BIND configuration file
/var/named/chroot/var/named               BIND zone files
```

system-config-bind

The system-config-bind tool provides a desktop interface for managing your DNS server. Choose System | Administration | Server Settings | Domain Name System to start it. The DNS server and its zone files will be listed (see Figure 30-1).

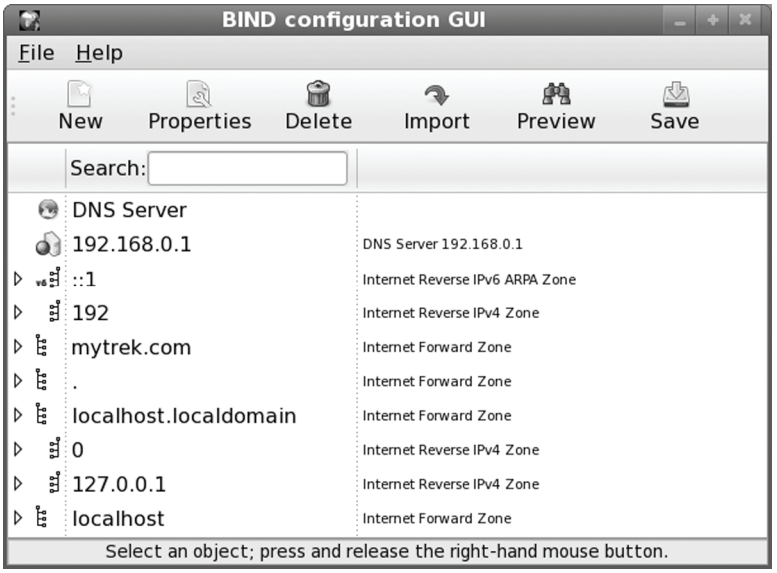


Figure 30-2: system-config-bind

To add a zone, click on the DNS server entry and click the New button. From the list select Zone. You can also right-click on the DNS server entry. This opens a New Zone window with menus to select the class, origin type, and the zone type (see Figure 30-3). The zone type can be master, slave or forward, among others. The class is usually IN. The origin type can be forward or reverse mapping for IPv4 or IPv6 protocols. For the class and origin type, once you select an entry, click their OK buttons.

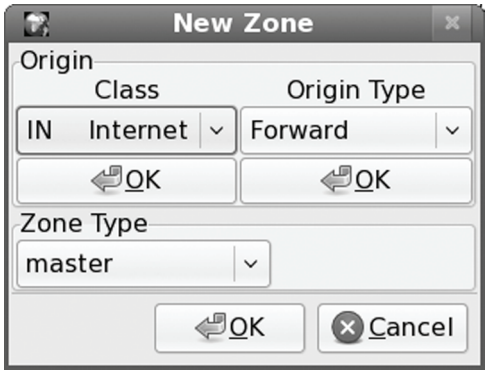


Figure 30-3: system-config-bind, new zone class and origin

When you click the OK button for the Origin Type, then text box opens where you can enter the zone domain name. Be sure to place a period at the end (see Figure 30-4).

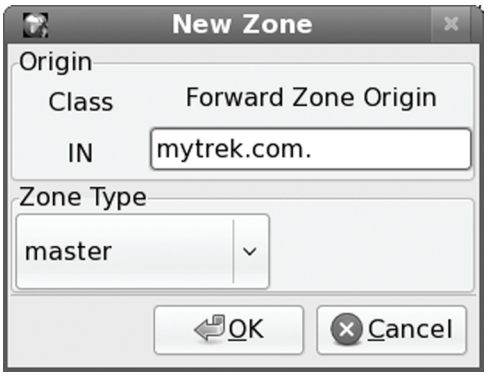


Figure 30-4: system-config-bind, new zone domain and type

After you click OK, the zone is added to DNS server listing. The appropriate zone file will be created in **/var/named/chroot/var/named**.

Once you have created a zone file, you can add records to it by right-clicking on its entry and selecting the kind of record you want from the Add submenu (see Figure 30-5). You can also select the zone entry and click the New button. You then enter the record information.

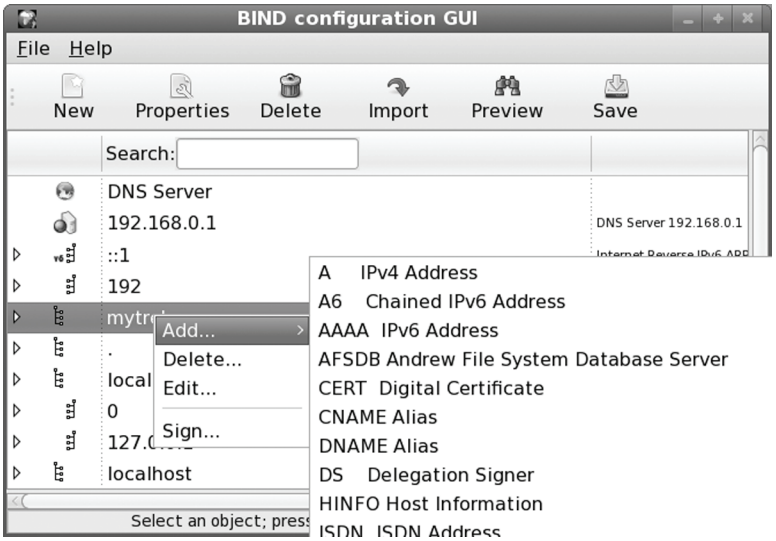


Figure 30-5: system-config-bind, select a record type

To see the actual files that system-config-bind is generating, click the Preview button. This lists all the files set up for your server. Click on the file to display its actual contents. The first entry will be the **/etc/named.conf** file, followed by zone files. Changes are not made until you click the Save button. You are first asked to confirm. Your current files will be replaced.

The system-config-bind tool can become complicated to use. It supports all the different kinds of records and zone files, including security entries. See the system-config-bind manual for a detailed discussion of all its features. Click Manual in the Help menu.

named.conf

The configuration file for the **named** daemon is **named.conf**. A link to it is located in the **/etc** directory (**/etc/named.conf**), with the original file located in the **/var/named/chroot/etc** directory. It uses a flexible syntax similar to C programs. The format enables easy configuration of selected zones, enabling features such as access control lists and categorized logging. The **named.conf** file consists of BIND configuration statements with attached blocks within which specific options are listed. A configuration statement is followed by arguments and a block that is delimited with braces. Within the block are lines of option and feature entries. Each entry is terminated with a semicolon. Comments can use the C, C++, or Shell/Perl syntax: enclosing **/* */**, preceding **//**, or preceding **#**. The following example shows a **zone** statement followed by the zone name and a block of options that begin with an opening brace (**{**). Each option entry ends with a semicolon. The entire block ends with a closing brace, also followed by a semicolon. The format for a **named.conf** entry is show here, along with the different kinds of comments allowed. Tables 35-5, 35-6, and 35-7 list several commonly used statements and options.

```
// comments
/* comments */
# comments

statements {
 options and features; //comments
};
```

The following example shows a simple caching server entry:

```
// a caching only nameserver config
//
zone "." {
       type hint;
       file "named.ca";
       };
```

Once you have created your configuration file, you should check its syntax with the **named-checkconfig** tool. Enter the command on a shell command line. If you do not specify a configuration file, it will default to **/etc/named.conf**.

```
named-checkconfig
```

The zone Statement

The **zone** statement is used to specify the domains the name server will service. You enter the keyword **zone**, followed by the name of the domain placed within double quotes. Do not place a period at the end of the domain name. In the following example, a period is within the domain name, but not at the end, "**mytrek.com**"; this differs from the zone file, which requires a period at the end of a complete domain name.

Type	Description
master	Primary DNS zone
slave	Slave DNS server; controlled by a master DNS server
hint	Set of root DNS Internet servers
forward	Forwards any queries in it to other servers
stub	Like a slave zone, but holds only names of DNS servers

Table 30-3: DNS BIND Zone Types

Statement	Description
`/* comment */`	BIND comment in C syntax.
`// comment`	BIND comment in C++ syntax.
`# comment`	BIND comment in Unix shell and Perl syntax.
`acl`	Defines a named IP address matching list.
`include`	Includes a file, interpreting it as part of the **named.conf** file.
`key`	Specifies key information for use in authentication and authorization.
`logging`	Specifies what the server logs and where the log messages are sent.
`options`	Global server configuration options and defaults for other statements.
`controls`	Declares control channels to be used by the ndc utility.
`server`	Sets certain configuration options for the specified server basis.
`sortlists`	Gives preference to specified networks according to a queries source.
`trusted-keys`	Defines DNSSEC keys preconfigured into the server and implicitly trusted.
`zone`	Defines a zone.
`view`	Defines a view.

Table 30-4: BIND Configuration Statements

After the zone name, you can specify the class **`in`**, which stands for Internet. You can also leave it out, in which case **`in`** is assumed (there are only a few other esoteric classes that are rarely used). Within the zone block, you can place several options (see Table 30-3). Two essential options are **`type`** and **`file`**. The **`type`** option is used to specify the zone's type. The **`file`** option is used to specify the name of the zone file to be used for this zone. You can choose from several types of zones: master, slave, stub, forward, and hint. *Master* specifies that the zone holds master information and is authorized to act on it. A master server was called a primary server in the older 4.*x* BIND configuration. *Slave* indicates that the zone needs to update its data periodically from a specified master name server. You use this entry if your name server is operating as a secondary server for another primary (master) DNS server. A *stub zone* copies only other name server entries, instead of the entire zone. A *forward zone* directs all queries to name servers specified in a

forwarders statement. A *hint zone* specifies the set of root name servers used by all Internet DNS servers. You can also specify several options that can override any global options set with the **options** statement. Table 30-3 lists the BIND zone types. The following example shows a simple **zone** statement for the **mytrek.com** domain. Its class is Internet (in), and its type is master. The name of its zone file is usually the same as the zone name with the **.db** extension, in this case, "**mytrek.com.db**."

```
zone "mytrek.com" in {
       type master;
       file "mytrek.com.db";
       };
```

Configuration Statements

Other statements, such as **acl**, **server**, **options**, and **logging**, enable you to configure different features for your name server (see Table 30-4). The **server** statement defines the characteristics to be associated with a remote name server, such as the transfer method and key ID for transaction security. The **control** statement defines special control channels. The **key** statement defines a key ID to be used in a **server** statement that associates an authentication method with a particular name server (see "DNSSEC" later in this chapter). The **logging** statement is used to configure logging options for the name server, such as the maximum size of the log file and a severity level for messages. Table 30-5 lists the BIND statements. The **sortlists** statement lets you specify preferences to be used when a query returns multiple responses. For example, you could give preference to your localhost network or to a private local network such a 192.168.0.0.

Option	Description
type	Specifies a zone type.
file	Specifies the zone file for the zone.
directory	Specifies a directory for zone files.
forwarders	Lists hosts for DNS servers where requests are to be forwarded.
masters	Lists hosts for DNS master servers for a slave server.
notify	Allows master servers to notify their slave servers when the master zone data changes and updates are needed.
allow-transfer	Specifies which hosts are allowed to receive zone transfers.
allow-query	Specifies hosts that are allowed to make queries.
allow-recursion	Specifies hosts that are allowed to perform recursive queries on the server.

Table 30-5: Zone Options

The options Statement

The **options** statement defines global options and can be used only once in the configuration file. An extensive number of options cover such components as forwarding, name

checking, directory path names, access control, and zone transfers, among others (see Table 30-6). A complete listing can be found in the BIND documentation.

The directory Option

A critically important option found in most configuration files is the **directory** option, which holds the location of the name server's zone and cache files on your system. The following example is taken from the **/etc/named.conf** file. This example specifies the zone files are located in the **/var/named** directory. In this directory, you can find your zone files, including those used for your local system. The example uses IPv4 addresses.

```
options {
        directory "/var/named";
        forwarders { 192.168.0.34;
                     192.168.0.47;
                   };
       };
```

The forwarders Option

Another commonly used global option is the **forwarders** option. With the **forwarders** option, you can list several DNS servers to which queries can be forwarded if they cannot be resolved by the local DNS server. This is helpful for local networks that may need to use a DNS server connected to the Internet. The **forwarders** option can also be placed in forward zone entries.

Option	Description
sortlist	Gives preference to specified networks according to a queries source.
directory	Specifies a directory for zone files.
forwarders	Lists hosts for DNS servers where requests are to be forwarded.
allow-transfer	Specifies which hosts are allowed to receive zone transfers.
allow-query	Specifies hosts that are allowed to make queries.
allow-recursion	Specifies hosts that are allowed to perform recursive queries on the server.
notify	Allows master servers to notify their slave servers when the master zone data changes and updates are needed.
blackhole	Option to eliminate denial response by **allow-query**.
recursion	Allow the DNS server to query other DNS servers it does not control.

Table 30-6: Bind Options for the options Statement

The notify Option

With the **notify** option turned on, the master zone DNS servers send messages to any slave DNS servers whenever their configuration has changed. The slave servers can then perform zone transfers in which they download the changed configuration files. Slave servers always use the DNS configuration files copied from their master DNS servers. The **notify** option takes one argument, **yes** or **no**, where **yes** is the default. With the **no** argument, you can have the master server not send out any messages to the slave servers, in effect preventing any zone transfers.

The IPv6 reverse mapping uses the IP6.ARPA domain in the reverse zone entries). IP6.ARPA uses bit labels providing bit-level specification for the address. This is simply the full hexadecimal address, including zeros, without intervening colons. You can also use the nibble format, separating each numeral with a period.

An named.conf Example

The following example is a simple **named.conf** file based on the example provided in the BIND documentation. This example shows samples of several of the configuration statements. The file begins with comments using C++ syntax, **//**. The **options** statement has a directory entry that sets the directory for the zone and cache files to **/var/named**. Here, you find links to your zone files, such as **named.local** and reverse mapping files, along with the cache file, **named.ca**. The original files will be located in **/var/named/chroot/var/named**. The first **zone** statement (**.**) defines a hint zone specifying the root name servers. The cache file listing these servers is **named.ca**. The second **zone** statement defines a zone for the **mytrek.com** domain. Its type is master, and its zone file is named "**mytrek.com**."

The next zone is used for reverse IPv4 mapping of the previous zone. Its name is made up of a reverse listing of the **mytrek.com** domain's IP address with the term **in-addr.arpa** appended. The domain address for **mytrek.com** is 192.168.0, so the reverse is 1.168.192. The **in-addr.arpa** domain is a special domain that supports gateway location and Internet address–to–host mapping. The **include** statement include the RFC 1012 zone definitions for the loopback interface, the method used by the system to address itself and enable communication between local users on the system.

The last statement will include the named.rfc1912.zones configuration file which defines zone statements for the localhost and its reverse lookup.

```
include "/etc/named.rfc1912.zones"
```

Caching-Only Server

When BIND is initially installed, it creates a default configuration for what is known as a caching-only server. A *caching-only* server copies queries made by users and saves them in a cache, for use later if the queries are repeated. This can save DNS lookup response times. The cache is held in memory and lasts only as long as **named** runs. The caching-only server configuration file is named.caching-nameserver.conf file. This file contains basic options for managing the caching server. It includes the **named.**rfc1912.zones file (also in **/etc**) which contains the zone configurations for the caching server.

named.conf

```
//
// A simple BIND 9 configuration
//

logging {
         category cname { null; };
         };

options {
         directory "/var/named";
        };

zone "." {
          type hint;
          file "named.ca";
         };

zone "mytrek.com" {
             type master;
             file "mytrek.com";
             };
zone "1.168.192.in-addr.arpa" {
                             type master;
                             file "192.168.0.db";
                            };

zone "\[xFC00000000000000/64].IP6.ARPA" {
                             type master;
                             file "fec.ip6.arpa.db";
                            };


include "/etc/named.rfc1912.zones"
```

Resource Records for Zone Files

Your name server holds domain name information about the hosts on your network in resource records placed in zone and reverse mapping files. Resource records are used to associate IP addresses with fully qualified domain names. You need a record for every computer in the zone that the name server services. A record takes up one line, though you can use parentheses to use several lines for a record, as is usually the case with SOA records. A resource record uses the Standard Resource Record Format as shown here:

```
name [<ttl>] [<class>] <type> <rdata> [<comment>]
```

Here, *name* is the name for this record. It can be a domain name or a hostname (fully qualified domain name). If you specify only the hostname, the default domain is appended. If no name entry exists, the last specific name is used. If the `@` symbol is used, the name server's domain name is used. *ttl* (time to live) is an optional entry that specifies how long the record is to be cached ($TTL directive sets default). *class* is the class of the record. The class used in most resource record

entries is IN, for Internet. By default, it is the same as that specified for the domain in the **named.conf** file. *type* is the type of the record. *rdata* is the resource record data. The following is an example of a resource record entry. The name is **rabbit.mytrek.com**, the class is Internet (IN), the type is a host address record (A), and the data is the IP address 192.168.0.2.

```
rabbit.mytrek.com.    IN   A   192.168.0.2
```

Resource Record Types

Different types of resource records exist for different kinds of hosts and name server operations (see Table 30-7 for a listing of resource record types). A, NS, MX, PTR, and CNAME are the types commonly used. A is used for host address records that match domain names with IP addresses. NS is used to reference a name server. MX specifies the host address of the mail server that services this zone. The name server has mail messages sent to that host. The PTR type is used for records that point to other resource records and is used for reverse mapping. CNAME is used to identify an alias for a host on your system.

```
$TTL 86400
```

Type	Description
A	An IPv4 host address, maps hostname to IPv4 address
AAAA	An IPv6 host address
A6	A chained IPv6 address
NS	Authoritative name server for this zone
CNAME	Canonical name, used to define an alias for a hostname
SOA	Start of Authority, starts DNS entries in zone file, specifies name server for domain, and other features such as server contact and serial number
WKS	Well-known service description
PTR	Pointer record, for performing reverse domain name lookups, maps IP address to hostname
RP	Text string that contains contact information about a host
HINFO	Host information
MINFO	Mailbox or mail list information
MX	Mail exchanger, informs remote site of your zone's mail server
TXT	Text strings, usually information about a host
KEY	Domain private key
SIG	Resource record signature
NXT	Next resource record

Table 30-7: Domain Name System Resource Record Types

Time To Live Directive and Field: $TTL

All zone files begin with a Time To Live directive, which specifies the time that a client should keep the provided DNS information before refreshing the information again from the DNS server. Realistically this should be at least a day, though if changes in the server are scheduled sooner, you can temporarily shorten the time, later restoring it. Each record, in fact, has a Time To Live value that can be explicitly indicated with the TTL field. This is the second field in a resource record. If no TTL field is specified in the record, then the default as defined by the $TLL directive can be used. The $TTL directive is placed at the beginning of each zone file. By default it will list the time in seconds, usually 86400, 24 hours.

You can also specify the time in days (d), hours (h), or minutes (m), as in

```
$TTL 2d3h
```

When used as a field, the TTL will be a time specified as the second field. In the following example, the turtle resource record can be cached for three days. This will override the default time in the TTL time directive:

```
turtle     3d      IN    A      192.168.0.1
```

Start of Authority: SOA

A zone or reverse mapping file always begins with a special resource record called the Start of Authority (SOA) record. This record specifies that all the following records are authoritative for this domain. It also holds information about the name server's domain, which is to be given to other name servers. An SOA record has the same format as other resource records, though its data segment is arranged differently. The format for an SOA record follows:

```
name {ttl} class SOA Origin Person-in-charge (
                                  Serial number
                                  Refresh
                                  Retry
                                  Expire
                                  Minimum )
```

Each zone has its own SOA record. The SOA begins with the zone name specified in the **named.conf** zone entry. This is usually a domain name. An **@** symbol is usually used for the name and acts like a macro expanding to the domain name. The *class* is usually the Internet class, IN. *SOA* is the type. *Origin* is the machine that is the origin of the records, usually the machine running your name server daemon. The *person-in-charge* is the e-mail address for the person managing the name server (use dots, not **@**, for the e-mail address, as this symbol is used for the domain name). Several configuration entries are placed in a block delimited with braces. The first is the *serial number*. You change the serial number when you add or change records, so that it is updated by other servers. The serial number can be any number, as long as it is incremented each time a change is made to any record in the zone. A common practice is to use the year-month-day-number for the serial number, where number is the number of changes in that day. For example, 1999120403 would be the year 1999, December 4, for the third change. Be sure to update it when making changes.

Refresh specifies the time interval for refreshing SOA information. *Retry* is the frequency for trying to contact an authoritative server. *Expire* is the length of time a secondary name server keeps information about a zone without updating it. *Minimum* is the length of time records in a zone live. The times are specified in the number of seconds.

The following example shows an SOA record. The machine running the name server is **turtle.mytrek.com,** and the e-mail address of the person responsible for the server is **hostmaster.turtle.mytrek.com**. Notice the periods at the ends of these names. For names with no periods, the domain name is appended. **turtle** would be the same as **turtle.mytrek.com**. When entering full hostnames, be sure to add the period so that the domain is not appended.

```
@ IN SOA turtle.mytrek.com. hostmaster.turtle.mytrek.com. (
                                    1997022700 ; Serial
                                    28800 ; Refresh
                                    14400 ; Retry
                                    3600000 ; Expire
                                    86400 ) ; Minimum
```

Name Server: NS

The name server record specifies the name of the name server for this zone. These have a resource record type of NS. If you have more than one name server, list them in NS records. These records usually follow the SOA record. As they usually apply to the same domain as the SOA record, their name field is often left blank to inherit the server's domain name specified by the **@** symbol in the previous SOA record.

```
              IN   NS      turtle.mytrek.com.
```

You can, if you wish, enter the domain name explicitly as shown here:

```
mytrek.com.  IN   NS      turtle.mytrek.com.
```

Address Record: A, A6, and AAAA

Resource records of type A are address records that associate a fully qualified domain name with an IP address. Often, only their hostname is specified. Any domain names without a terminating period automatically have the domain appended to them. Given the domain **mytrek.com**, the **turtle** name in the following example is expanded to **turtle.mytrek.com**:

```
rabbit.mytrek.com. IN    A      192.168.0.2
turtle             IN    A      192.168.0.1
```

BIND supports IPv6 addresses. IPv6 IP addresses have a very different format from that of the IPv4 addresses commonly used. Instead of the numerals arranged in four segments, IPv6 uses hexadecimal numbers arranged in seven segments. In the following example, **turtle.mytrek.com** is associated with a unique-local IPv6 address: **fc00::**. Recall that there are only three fields in a unique-local address: format prefix, subnet identifier, and interface identifier. The empty segments of the subnet identifier can be represented by an empty colon pair (::). The interface identifier follows, **8:800:200C:417A**.

```
turtle.mytrek.com. IN    AAAA    FC00::8:800:200C:417A
```

IPv6 also supports the use of IPv4 addresses as an interface identifier, instead of the MAC-derived identifier. The network information part of the IPv6 address would use IPv6 notation, and the remaining interface (host) identifier would use the full IPv4 address. These are known as mixed addresses. In the next example, **lizard.mytrek.com** is given a mixed address using IPv6 network information and IPv4 interface information. The IPv6 network information is for an IPv6 unique-local address.

```
lizard.mytrek.com. IN      AAAA        fc00::192.168.0.3
```

The AAAA record is used in most networks for an IPv6 record. An AAAA record operates much like a standard A address record, requiring a full IPv6 address. An A6 record is an experimental version of the IPv6 record. It can be more flexible, in that it does not require a full address. Instead you chain A6 records together, specifying just part of the address in each. For example, you could specify just an interface identifier for a host, letting the network information be provided by another IPv6 record (you can implement an A6 record with a full address, just like an AAAA record). In the next example, the first A6 record lists only the address for the interface identifier for the host **divit**. Following the address is the domain name, **mytrek.com**, whose address is to be used to complete **divit**'s address, providing network information. The next A6 record provides the network address information for **mytrek.com**.

```
divit.mygolf.com. IN    A6   0:0:0:0:1234:5678:3466:af1f  mytrek.com.
mytrek.com.       IN    A6    3ffe:8050:201:1860::
```

Mail Exchanger: MX

The Mail Exchanger record, MX, specifies the mail server that is used for this zone or for a particular host. The mail exchanger is the server to which mail for the host is sent. In the following example, the mail server is specified as **turtle.mytrek.com**. Any mail sent to the address for any machines in that zone will be sent to the mail server, which in turn will send it to the specific machines. For example, mail sent to a user on **rabbit.mytrek.com** will first be sent to **turtle.mytrek.com**, which will then send it on to **rabbit.mytrek.com**. In the following example, the host 192.168.0.1 (**turtle.mytrek.com**) is defined as the mail server for the **mytrek.com** domain:

```
mytrek.com. IN     MX   10   turtle.mytrek.com.
```

You could also inherit the domain name from the SOA record, leaving the domain name entry blank.

```
             IN     MX   turtle.mytrek.com.
```

You could use the IP address instead, but in larger networks, the domain name may be needed to search for and resolve the IP address of a particular machine, which could change.

```
mytrek.com. IN     MX   10   192.168.0.1
```

An MX record recognizes an additional field that specifies the ranking for a mail exchanger. If your zone has several mail servers, you can assign them different rankings in their MX records. The smaller number has a higher ranking. This way, if mail cannot reach the first mail server, it can be routed to an alternate server to reach the host. In the following example, mail for hosts on the **mytrek.com** domain is first routed to the mail server at 192.168.0.1 (**turtle.mytrek.com**), and if that fails, it is routed to the mail server at 192.168.0.2 (**rabbit.mytrek.com**).

```
mytrek.com. IN MX 10 turtle.mytrek.com.
            IN MX 20 rabbit.mytrek.com.
```

You can also specify a mail server for a particular host. In the following example, the mail server for **lizard.mytrek.com** is specified as **rabbit.mytrek.com**:

```
lizard.mytrek.com. IN      A       192.168.0.3
                   IN      MX  10   rabbit.mytrek.com.
```

Aliases: CNAME

Resource records of type CNAME are used to specify alias names for a host in the zone. Aliases are often used for machines running several different types of servers, such as both Web and FTP servers. They are also used to locate a host when it changes its name. The old name becomes an alias for the new name. In the following example, **ftp.mytrek.com** is an alias for a machine actually called **turtle.mytrek.com**:

```
ftp.mytrek.com. IN CNAME turtle.mytrek.com.
```

The term CNAME stands for canonical name. The canonical name is the actual name of the host. In the preceding example, the canonical name is **turtle.mytrek.com**. The alias, also known as the CNAME, is **ftp.mytrek.com**. In a CNAME entry, the alias points to the canonical name. Aliases cannot be used for NS (name server) or MX (mail server) entries. For those records, you need to use the original domain name or IP address.

A more stable way to implement aliases is simply to create another address record for a host or domain. You can have as many hostnames for the same IP address as you want, provided they are certified. For example, to make **www.mytrek.com** an alias for **turtle.mytrek.com**, you only have to add another address record for it, giving it the same IP address as **turtle.mytrek.com**.

```
turtle.mytrek.com. IN A 192.168.0.1
www.mytrek.com. IN A 192.168.0.1
```

Pointer Record: PTR

A PTR record is used to perform reverse mapping from an IP address to a host. PTR records are used in the reverse mapping files. The name entry holds a reversed IP address, and the data entry holds the name of the host. The following example maps the IP address 192.168.0.1 to **turtle.mytrek.com**:

```
1.1.168.192 IN PTR turtle.mytrek.com.
```

In a PTR record, you can specify just that last number segment of the address (the host address) and let DNS fill in the domain part of the address. In the next example, 1 has the domain address, 1.168.192, automatically added to give 1.1.168.192:

```
1 IN PTR turtle.mytrek.com.
```

Host Information: HINFO, RP, MINFO, and TXT

The HINFO, RP, MINFO, and TXT records are used to provide information about the host. The RP record enables you to specify the person responsible for a certain host. The HINFO record provides basic hardware and operating system identification. The TXT record is used to

enter any text you want. MINFO provides a host's mail and mailbox information. These are used sparingly, as they may give too much information out about the server.

Zone Files

A DNS server uses several zone files covering different components of the DNS. Each zone uses two zone files: the principal zone file and a reverse mapping zone file. The *zone file* contains the resource records for hosts in the zone. A *reverse mapping file* contains records that provide reverse mapping of your domain name entries, enabling you to map from IP addresses to domain names. The name of the file used for the zone file can be any name. The name of the file is specified in the **zone** statement's file entry in the **named.conf** file. If your server supports several zones, you may want to use a name that denotes the specific zone. Most systems use the domain name as the name of the zone file. For example, the zone **mytrek.com** would have a zone file with the same name and the extension **db**, as in **mytrek.com.db**. The zone file used in the following example is called **mytrek.com.db**. The reverse mapping file can also be any name, though it is usually the reverse IP address domain specified in its corresponding zone file. For example, in the case of **mytrek.com.db** zone file, the reverse mapping file might be called **192.168.0.db**, the IP address of the **mytrek.com** domain defined in the **mytrek.com.db** zone file. This file would contain reverse mapping of all the host addresses in the domain, allowing their hostname addresses to be mapped to their corresponding IP addresses. In addition, BIND sets up a cache file and a zone files for the localhost. The cache file holds the resource records for the root name servers to which your name server connects. The cache file can be any name, although it is usually called **named.ca**. The localhost zone files holds reverse IP resource records for the local loopback interface, localhost. On Fedora, the localhost file is **named.localhost**. and its reverse mapping file is **named.loopback**. The localhost zone is configured by statements in the **named.rfc1912.zones** configuration file which is included in the **named.conf** file. The localhost addresses are configured according to the RFC 1912 Internet Protocol standard.

Once you have created your zone files, you should check their syntax with the **named-checkzone** tool. This tool requires that you specify both a zone and a zone file. In the following example, in the **/var/named** directory, the zone **mytrek.com** in the zone file **mytrek.com.db** is checked:

```
named-checkzone  mytrek.com mytrek.com.db
```

Zone Files for Internet Zones

A zone file holds resource records that follow a certain format. The file begins with general directives to define default domains or to include other resource record files. These are followed by a single SOA record, name server and domain resource records, and then resource records for the different hosts. Comments begin with a semicolon and can be placed throughout the file. The **@** symbol operates like a special macro, representing the domain name of the zone to which the records apply. The **@** symbol is used in the first field of a resource or SOA record as the zone's domain name. Multiple names can be specified using the ***** matching character. The first field in a resource record is the name of the domain to which it applies. If the name is left blank, the previous explicit name entry in another resource record is automatically used. This way, you can list several entries that apply to the same host without having to repeat the hostname. Any host or domain name used throughout this file that is not terminated with a period has the zone's domain appended to it. For example, if the zone's domain is **mytrek.com** and a resource record has only

the name **rabbit** with no trailing period, the zone's domain is automatically appended to it, giving you **rabbit.mytrek.com.**. Be sure to include the trailing period whenever you enter the complete fully qualified domain name, **turtle.mytrek.com.**, for example.

Directives

You can also use several directives to set global attributes. $ORIGIN sets a default domain name to append to address names that do not end in a period. $INCLUDE includes a file. $GENERATE can generate records whose domain or IP addresses differ only by an iterated number. The $ORIGIN directive is often used to specify the root domain to use in address records. Be sure to include the trailing period. The following example sets the domain origin to **mytrek.com** and will be automatically appended to the **lizard** host name that follows:

```
$ORIGIN   mytrek.com.
lizard   IN   A    192.168.0.2
```

SOA Record

A zone file begins with an SOA record specifying the machine the name server is running on, among other specifications. The **@** symbol is used for the name of the SOA record, denoting the zone's domain name. After the SOA, the name server resource records (NS) are listed. Just below the name server records are resource records for the domain itself. Resource records for host addresses (A), aliases (CNAME), and mail exchangers (MX) follow. The following example shows a sample zone file, which begins with an SOA record and is followed by an NS record, resource records for the domain, and then resource records for individual hosts:

```
; Authoritative data for turle.mytrek.com
;
$TTL 86400
@ IN SOA turtle.mytrek.com. hostmaster.turtle.mytrek.com.(
                              93071200 ; Serial number
                                 10800 ; Refresh 3 hours
                                  3600 ; Retry 1 hour
                                3600000 ; Expire 1000 hours
                                86400 ) ; Minimum 24 hours

             IN      NS        turtle.mytrek.com.
             IN      A         192.168.0.1
             IN      MX   10   turtle.mytrek.com.
             IN      MX   15   rabbit.mytrek.com.

turtle       IN      A         192.168.0.1
             IN      HINFO     PC-686 LINUX
gopher       IN      CNAME     turtle.mytrek.com.
ftp          IN      CNAME     turtle.mytrek.com.
www          IN      A         192.168.0.1

rabbit       IN      A         192.168.0.2

lizard       IN      A         192.168.0.3
             IN      HINFO     MAC MACOS
```

The first two lines are comments about the server for which this zone file is used. Notice that the first two lines begin with a semicolon. The class for each of the resource records in this file is IN, indicating these are Internet records. The SOA record begins with an **@** symbol that stands for the zone's domain. In this example, it is **mytrek.com**. Any host or domain name used throughout this file that is not terminated with a period has this domain appended to it. For example, in the following resource record, **turtle** has no period, so it automatically expands to **turtle.mytrek.com**. The same happens for **rabbit** and **lizard**. These are read as **rabbit.mytrek.com** and **lizard.mytrek.com**. Also, in the SOA, notice that the e-mail address for hostmaster uses a period instead of an **@** symbol; **@** is a special symbol in zone files and cannot be used for any other purpose.

Nameserver Record

The next resource record specifies the name server for this zone. Here, it is **mytrek.com.**. Notice the name for this resource record is blank. If the name is blank, a resource record inherits the name from the previous record. In this case, the NS record inherits the value of **@** in the SOA record, its previous record. This is the zone's domain, and the NS record specifies **turtle.mytrek.com** as the name server for this zone.

```
            IN   NS    turtle.mytrek.com.
```

Here the domain name is inherited. The entry can be read as the following. Notice the trailing period at the end of the domain name:

```
mytrek.com. IN   NS    turtle.mytrek.com.
```

Address Record

The following address records set up an address for the domain itself. This is often the same as the name server, in this case 192.168.0.1 (the IP address of **turtle.mytrek.com**). This enables users to reference the domain itself, rather than a particular host in it. A mail exchanger record follows that routes mail for the domain to the name server. Users can send mail to the **mytrek.com** domain and it will be routed to **turtle.mytrek.com**.

```
              IN    A     192.168.0.1
```

Here the domain name is inherited. The entry can be read as the following:

```
mytrek.com. IN   A     192.168.0.1
```

Mail Exchanger Record

The next records are mail exchanger (MX) records listing **turtle.mytrek.com** and **fast.mytrek.com** as holding the mail servers for this zone. You can have more than one mail exchanger record for a host. More than one host may exist through which mail can be routed. These can be listed in mail exchanger records for which you can set priority rankings (a smaller number ranks higher). In this example, if **turtle.mytrek.com** cannot be reached, its mail is routed through **rabbit.mytrek.com**, which has been set up also to handle mail for the **mytrek.com** domain:

```
          IN     MX    100    turtle.mytrek.com.
          IN     MX    150    rabbit.mytrek.com.
```

Again the domain name is inherited. The entries can be read as the following:

```
mytrek.com.    IN      MX  100   turtle.mytrek.com.
mytrek.com.    IN      MX  150   rabbit.mytrek.com.
```

Address Record with Host Name

The following resource record is an address record (A) that associates an IP address with the fully qualified domain name **turtle.mytrek.com**. The resource record name holds only **turtle** with no trailing period, so it is automatically expanded to **turtle.mytrek.com**. This record provides the IP address to which **turtle.mytrek.com** can be mapped.

```
turtle   IN    A     192.168.0.1
```

Inherited Names

Several resource records immediately follow that have blank names. These inherit their names from the preceding full record—in this case, **turtle.mytrek.com**. In effect, these records also apply to that host. Using blank names is an easy way to list additional resource records for the same host (notice that an apparent indent occurs). The first record is an information record, providing the hardware and operating system for the machine.

```
         IN    HINFO    PC-686 LINUX
```

Alias Records

If you are using the same machine to run several different servers, such as Web, FTP, and Gopher servers, you may want to assign aliases to these servers to make accessing them easier for users. Instead of using the actual domain name, such as **turtle.mytrek.com**, to access the Web server running on it, users may find using the following is easier: for the Web server, **www.mytrek.com**; and for the FTP server, **ftp.mytrek.com**. In the DNS, you can implement such a feature using alias records. In the example zone file, two CNAME alias records exist for the **turtle.mytrek.com** machine: FTP and Gopher. The next record implements an alias for **www** using another address record for the same machine. None of the name entries ends in a period, so they are appended automatically with the domain name **mytrek.com**. **www.mytrek.com**, **ftp.mytrek.com**, and **gopher.mytrek.com** are all aliases for **turtle.mytrek.com**. Users entering those URLs automatically access the respective servers on the **turtle.mytrek.com** machine.

Address and mail exchanger records are then listed for the two other machines in this zone: **rabbit.mytrek.com** and **lizard.mytrek.com**. You could add HINFO, TXT, MINFO, or alias records for these entries.

IPv6 Zone File Example

This is the same zone file using IPv6 addresses. The addresses are unique-local (FC00), instead of global (3), providing private network addressing. The loopback device is represented by the IPv6 address `::1`. The AAAA IPv6 address records are used.

```
; Authoritative data for turle.mytrek.com, IPv6 version
;
$TTL 1d
@ IN SOA turtle.mytrek.com. hostmaster.turtle.mytrek.com.(
                             93071200 ; Serial number
                                10800 ; Refresh 3 hours
                                 3600 ; Retry 1 hour
                              3600000 ; Expire 1000 hours
                              86400 ) ; Minimum 24 hours

            IN      NS        turtle.mytrek.com.
            IN      AAAA        FC00::8:800:200C:417A
            IN      MX   10   turtle.mytrek.com.
            IN      MX   15   rabbit.mytrek.com.

turtle      IN      AAAA        FC00::8:800:200C:417A
            IN      HINFO     PC-686 LINUX
gopher      IN      CNAME     turtle.mytrek.com.
ftp         IN      CNAME     turtle.mytrek.com.
www         IN      AAAA        FC00::8:800:200C:417A

rabbit      IN      AAAA        FC00::FEDC:BA98:7654:3210

lizard      IN      AAAA        FC00::E0:18F7:3466:7D
            IN      HINFO     MAC MACOS
```

Reverse Mapping File

Reverse name lookups are enabled using a reverse mapping file. *Reverse mapping* files map fully qualified domain names to IP addresses. This reverse lookup capability is unnecessary, but it is convenient to have. With reverse mapping, when users access remote hosts, their domain name addresses can be used to identify their own host, instead of only the IP address. The name of the file can be anything you want. On most current distributions, it is the zone's domain address (the network part of a zone's IP address). For example, the reverse mapping file for a zone with the IP address of 192.168.0.1 is 192.168.0. Its full pathname would be something like **/var/named/192.168.0.db**. On some systems using older implementations of BIND, the reverse mapping filename may consist of the root name of the zone file with the extension **.rev**. For example, if the zone file is called **mytrek.com**, the reverse mapping file would be called something like **mytrek.rev**.

IPv4 in-addr.arpa Reverse Mapping Format

In IPv4, the zone entry for a reverse mapping in the **named.conf** file uses a special domain name consisting of the IP address in reverse, with an **in-addr.arpa** extension. This reverse IP address becomes the zone domain referenced by the **@** symbol in the reverse mapping file. For example, the reverse mapping zone name for a domain with the IP address of **192.168.43** would be **43.168.192.in-addr.arpa**. In the following example, the reverse domain name for the domain address **192.168.0** is **1.168.192.in-addr.arpa**:

```
zone "1.168.192.in-addr.arpa" in {
          type master;
          file "192.168.0.db";
          };
```

A reverse mapping file begins with an SOA record, which is the same as that used in a forward mapping file. Resource records for each machine defined in the forward mapping file then follow. These resource records are PTR records that point to hosts in the zone. These must be actual hosts, not aliases defined with CNAME records. Records for reverse mapping begin with a reversed IP address. Each segment in the IP address is sequentially reversed. Each segment begins with the host ID, followed by reversed network numbers.

```
; reverse mapping of domain names 1.168.192.in-addr.arpa
;
$TTL 86400
@ IN SOA turtle.mytrek.com. hostmaster.turtle.mytrek.com.(
                         92050300 ; Serial (yymmddxx format)
                            10800 ; Refresh 3hHours
                             3600 ; Retry 1 hour
                          3600000 ; Expire 1000 hours
                          86400 ) ; Minimum 24 hours

@             IN    NS       turtle.mytrek.com.
1             IN    PTR      turtle.mytrek.com.
2.1.168.192   IN    PTR      rabbit.mytrek.com.
3             IN    PTR      lizard.mytrek.com.
```

If you list only the host ID with no trailing period, the zone domain is automatically attached. In the case of a reverse mapping file, the zone domain as specified in the **zone** statement is the domain IP address backward. The 1 expands to 1.1.168.192. In the following example, **turtle** and **lizard** inherit the domain IP address, whereas **rabbit** has its address explicitly entered:

IPv6 IP6.ARPA Reverse Mapping Format

In IPv6, reverse mapping can be handled either with the current IP6.ARPA domain format, or with the older IP6.INT format. With IP6.ARPA, the address is represented by a bit-level representation that places the hexadecimal address within brackets. The first bracket is preceded by a backslash. The address must be preceded by an *x* indicating that it is a hexadecimal address. Following the address is a number indicating the number of bits referenced. In a 128-bit address, usually the first 64 bits reference the network address and the last 64 bits are for the interface address. The following example shows the network and interface addresses for lizard.

```
FC00:0000:0000:0000:00E0:18F7:3466:007D  lizard IPv6 address
\[xFC00000000000000/64]                  lizard network address
\[x00E018F73466007D/64]                  lizard interface address
```

The zone entry for a reverse mapping in the **named.conf** file with an **IP6.ARPA** extension would use the bit-level representation for the network address.

```
zone "\[xfc00000000000000/64].IP6.ARPA" in {
          type master;
          file "fec.ip6.arpa";
          };
```

A reverse mapping file then uses the same bit-level format for the interface addresses.

```
$TTL 1d
@ IN SOA turtle.mytrek.com. hostmaster.turtle.mytrek.com.(
                        92050300 ; Serial (yymmddxx format)
                           10800 ; Refresh 3hHours
                            3600 ; Retry 1 hour
                         3600000 ; Expire 1000 hours
                         86400 ) ; Minimum 24 hours

@                           IN    NS       turtle.mytrek.com.
\[x00080800200C417A/64]     IN    PTR      turtle.mytrek.com.
\[xFEDCBA9876543210/64]     IN    PTR      rabbit.mytrek.com.
\[x00E018F73466007D/64]     IN    PTR      lizard.mytrek.com.
```

IPv6 IP6.INT Reverse Mapping Format

The older IP6.INT format uses a nibble format for the IPv6 address. This has since been replaced by the IPv6.ARPA format. The hexadecimal address is segmented into each hex number, separated by a period and written in reverse. This gives you 32 hex numbers in reverse order. The IP6.INT version for the lizard address are shown here:

```
FC00:0000:0000:0000:00E0:18F7:3466:007D   lizard IPv6 address
0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.c.e.f     lizard network address
D.7.0.0.6.6.4.3.7.F.8.1.0.E.0.0           lizard interface address
```

The zone entry for a reverse mapping in the **named.conf** file with an **IP6.INT** extension would use the reverse nibble format for the network address.

```
zone "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.c.e.f.IP6.INT" in {
          type master;
          file "fec.ip6.int";
          };
```

The reverse zone file then uses the reverse nibble format for each interface address.

```
$TTL 1d
@ IN SOA turtle.mytrek.com. hostmaster.turtle.mytrek.com.(
                        92050300 ; Serial (yymmddxx format)
                           10800 ; Refresh 3hHours
                            3600 ; Retry 1 hour
                         3600000 ; Expire 1000 hours
                         86400 ) ; Minimum 24 hours
$ORIGIN 0.0.0.0.0.0.0.0.0.0.0.0.0.c.e.f  IN  NS  turtle.mytrek.com.
A.7.1.4.C.0.0.2.0.0.8.0.8.0.0.0   IN    PTR      turtle.mytrek.com.
0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F   IN    PTR      rabbit.mytrek.com.
D.7.0.0.6.6.4.3.7.F.8.1.0.E.0.0   IN    PTR      lizard.mytrek.com.
```

RFC 1912 private address management: localhost

On Red Hat and Fedora, the zones for private address space, including localhost, is handled by the RFC 1912 configuration file, **named.1912.zones**. This file is read directly by the named.conf file, including it as part of your BIND configuration. It defines zones for localhost, localhost.localdomain, and the localhost reverse mapping zones for both IPv4 and IPv6 addressing.

Localhost zone file: named.localhost

The **named.localhost** zone file implements mapping for the local loopback interface known as localhost. This file includes support for both for IPv4 and for IPv6 addressing. The IPv4 address for localhost is **127.0.0.1**, and the IPv6 address is **::1**. These are special addresses that functions as the local address for your machine. It allows a machine to address itself.

The IPv4 address has the type A and the address 127.0.0.1, whereas the IPv6 address has the type AAAA and the address ::1.

```
       A      127.0.0.1
       AAAA   ::1
```

The named.localhost zone file is shown here.

named.localhost

```
$TTL 1D
@      IN SOA @ rname.invalid. (
                                       0      ; serial
                                       1D     ; refresh
                                       1H     ; retry
                                       1W     ; expire
                                       3H )   ; minimum
       NS     @
       A      127.0.0.1
       AAAA   ::1
```

named.1912.zones

```
// named.rfc1912.zones:
//
// Provided by Red Hat caching-nameserver package
//
// ISC BIND named zone configuration for zones recommended by
// RFC 1912 section 4.1 : localhost TLDs and address zones
// and http://www.ietf.org/internet-drafts/draft-ietf-dnsop-default-local-zones-
02.txt
// (c)2007 R W Franks
//
// See /usr/share/doc/bind*/sample/ for example named configuration files.
//

zone "localhost.localdomain" IN {
       type master;
       file "named.localhost";
       allow-update { none; };
};

zone "localhost" IN {
       type master;
       file "named.localhost";
       allow-update { none; };
};

zone "1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa"
IN {
       type master;
       file "named.loopback";
       allow-update { none; };
};

zone "1.0.0.127.in-addr.arpa" IN {
       type master;
       file "named.loopback";
       allow-update { none; };
};

zone "0.in-addr.arpa" IN {
       type master;
       file "named.empty";
       allow-update { none; };
};
```

Localhost Reverse Mapping: named.loopback

A localhost reverse mapping file implements reverse mapping for the local loopback interface known as **localhost**, This file is given the name **named.loopback** It is referenced by both the IPv4 and IPv6 reverse mapping zone statements.

In the IPv4 **zone** statement for this file, the name of the zone is **0.0.127.in-addr.arpa**. The domain part of the IP address is entered in reverse order, with **in-addr.arpa** appended to it, **0.0.127.IN-addr.arpa**. The **named.1912.zones** entry is shown here:

```
zone "0.0.127.in-addr.arpa" {
           type master;
           file "named.loopback";
           allow-update { none; };
  };
```

For IPv6 zone statement, the Fedora implementation uses the nibble format for the reverse mapping address:

```
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
```

The IPv6 localhost zone statement is shown here:

```
zone "1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa"
IN {
       type master;
       file "named.loopback";
       allow-update { none; };
};
```

The **named.loopback** zone file supports both IPv4 and IPv6 addresses. The NS record specifies the name server localhost should use. This file has a PTR record that maps the IP address to the localhost. The contents of the **named.localhost** file are shown here. Notice the trailing periods for localhost:

```
$TTL 1D
@       IN SOA  @  rname.invalid. (
                        0   ; serial
                        1D  ; refresh
                        1H  ; retry
                        1W  ; expire
                        3H ) ; minimum
        NS          @
        PTR         localhost.
```

Subdomains and Slaves

Adding a subdomain to a DNS server is a simple matter of creating an additional master entry in the **named.conf** file, and then placing name server and authority entries for that subdomain in your primary DNS server's zone file. The subdomain, in turn, has its own zone file with its SOA record and entries listing hosts, which are part of its subdomain, including any of its own mail and news servers.

Subdomain Zones

The name for the subdomain could be a different name altogether or a name with the same suffix as the primary domain. In the following example, the subdomain is called **beach.mytrek.com**. It could just as easily be called **mybeach.com**. The name server to that domain

is on the host **crab.beach.mytrek.com**, in this example. Its IP address is 192.168.0.33, and its zone file is **beach.mytrek.com**. The **beach.mytrek.com** zone file holds DNS entries for all the hosts being serviced by this name server. The following example shows zone entries for its **named.conf**:

```
zone "beach.mytrek.com" {
          type master;
          file "beach.mytrek.com.db";
          };

zone "1.168.192.in-addr.arpa" {
          type master;
          file "192.168.0.db";
          };
```

Subdomain Records

On the primary DNS server, in the example **turtle.mytrek.com**, you would place entries in the master zone file to identify the subdomain server's host and designate it as a name server. Such entries are also known as *glue records.* In this example, you would place the following entries in the **mytrek.com** zone file on **turtle.mytrek.com**:

```
beach.mytrek.com.    IN    NS    beach.mytrek.com.
beach.mytrek.com.    IN    A     192.168.0.33.
```

URL references to hosts serviced by **beach.mytrek.com** can now be reached from any host serviced by **mytrek.com**, which does not need to maintain any information about the **beach.mytrek.com** hosts. It simply refers such URL references to the **beach.mytrek.com** name server.

Slave Servers

A slave DNS server is tied directly to a master DNS server and periodically receives DNS information from it. You use a master DNS server to configure its slave DNS servers automatically. Any changes you make to the master server are automatically transferred to its slave servers. This transfer of information is called a *zone transfer.* Zone transfers are automatically initiated whenever the slave zone's refresh time is reached or the slave server receives a notify message from the master. The *refresh time* is the second argument in the zone's SOA entry. A notify message is automatically sent by the master whenever changes are made to the master zone's configuration files and the **named** daemon is restarted. In effect, slave zones are automatically configured by the master zone, receiving the master zone's zone files and making them their own.

Slave Zones

Using the previous examples, suppose you want to set up a slave server on **rabbit.mytrek.com**. Zone entries, as shown in the following example, are set up in the **named.conf** configuration file for the slave DNS server on **rabbit.mytrek.com**. The slave server is operating in the same domain as the master, and so it has the same zone name, **mytrek.com**. Its SOA file is named **slave.mytrek.com.db**. The slave zone files will be located in the **slaves** subdirectory. The term "slave" in the filename is merely a convention that helps identify it as a slave server configuration file. The **masters** statement lists its master DNS server—in this case, 192.168.0.1. Whenever the slave needs to make a zone transfer, it transfers data from that master

DNS server. The entry for the reverse mapping file for this slave server lists its reverse mapping file as **slave.192.168.0.db**.

```
zone "mytrek.com" {
        type slave;
        file "slave.mytrek.com.db";
        masters { 192.168.0.1;
        };

zone "1.168.192.in-addr.arpa" {
         type slave;
         file "slave.192.168.0.db";
         masters { 192.168.0.1;
         };
```

Slave Records

On the master DNS server, the master SOA zone file has entries in it to identify the host that holds the slave DNS server and to designate it as a DNS server. In this example, you would place the following in the **mytrek.com** zone file:

```
        IN      NS      192.168.0.2
```

You would also place an entry for this name server in the **mytrek.com** reverse mapping file:

```
        IN      NS      192.168.0.2
```

Controlling Transfers

The master DNS server can control which slave servers can transfer zone information from it using the **allow-transfer** statement. Place the statement with the list of IP addresses for the slave servers to which you want to allow access. Also, the master DNS server should be sure its **notify** option is not disabled. The **notify** option is disabled by a "notify no" statement in the options or zone **named.conf** entries. Simply erase the "no" argument to enable notify.

Incremental Zone Transfers

With BIND versions 8.2.2 and 9.0, BIND now supports incremental zone transfers (IXFR). Previously, all the zone data would be replaced in an update, rather than changes such as the addition of a few resource records simply being edited in. With incremental zone transfers, a database of changes is maintained by the master zone. Then only the changes are transferred to the slave zone, which uses this information to update its own zone files. To implement incremental zone transfers, you have to turn on the **maintain-ixfr-base** option in the options section.

```
maintain-ixfr-base yes;
```

You can then use the **ixfr-base** option in a zone section specify a particular database file to hold changes.

```
ixfr-base "db.mytrek.com.ixfr";
```

IP Virtual Domains

IP-based virtual hosting allows more than one IP address to be used for a single machine. If a machine has two registered IP addresses, either one can be used to address the machine. If you want to treat the extra IP address as another host in your domain, you need only create an address record for it in your domain's zone file. The domain name for the host would be the same as your domain name. If you want to use a different domain name for the extra IP, however, you have to set up a virtual domain for it. This entails creating a new **zone** statement for it with its own zone file. For example, if the extra IP address is 192.168.0.42 and you want to give it the domain name **sail.com**, you must create a new **zone** statement for it in your **named.conf** file with a new zone file. The **zone** statement would look something like this. The zone file is called **sail.com**:

```
zone "sail.com" in {
          type master;
          file "sail.com.db";
          };
```

In the **sail.com** file, the name server name is **turtle.mytrek.com** and the e-mail address is **hostmaster@turtle.mytrek.com**. In the name server (NS) record, the name server is **turtle.mytrek.com.** This is the same machine using the original address that the name server is running as. **turtle.mytrek.com** is also the host that handles mail addressed to **sail.com** (MX). An address record then associates the extra IP address 192.168.0.42 with the **sail.com** domain name. A virtual host on this domain is then defined as **jib.sail.com**. Also, **www** and **ftp** aliases are created for that host, creating **www.sail.com** and **ftp.sail.com** virtual hosts.

```
; Authoritative data for sail.com
;
$TTL 1d
@ IN SOA turtle.mytrek.com. hostmaster.turtle.mytrek.com. (
                               93071200 ; Serial (yymmddxx)
                                  10800 ; Refresh 3 hours
                                   3600 ; Retry 1 hour
                                3600000 ; Expire 1000 hours
                                86400 ) ; Minimum 24 hours

        IN      NS         turtle.mytrek.com.
        IN      MX   10    turtle.mytrek.com.
        IN      A          192.168.0.42 ;address of the sail.com domain

jib     IN      A         192.168.0.42
www     IN      A         jib.sail.com.
ftp     IN      CNAME     jib.sail.com.
```

In your reverse mapping file (**/var/named/1.168.192**), add PTR records for any virtual domains.

```
42.1.168.192      IN     PTR     sail.com.
42.1.168.192      IN     PTR     jib.sail.com.
```

You also have to configure your network connection to listen for both IP addresses on your machine.

Cache File

The *cache file* is used to connect the DNS server to root servers on the Internet. The file can be any name. On many systems, the cache file is called **named.ca**. Other systems may call the cache file **named.cache** or **roots.hints**. The cache file is usually a standard file installed by your BIND software, which lists resource records for designated root servers for the Internet. You can obtain a current version of the **named.ca** file from the **http://rs.internic.net** FTP site. The following example shows sample entries taken from the **named.ca** file:

```
; formerly NS.INTERNIC.NET
;
. 3600000 IN NS A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET. 3600000 A 198.41.0.4
;
; formerly NS1.ISI.EDU
;
. 3600000 NS B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET. 3600000 A 128.9.0.107
```

If you are creating an isolated intranet, you need to create your own root DNS server until you connect to the Internet. In effect, you are creating a fake root server. This can be another server on your system pretending to be the root or the same name server.

Dynamic Update: DHCP and Journal Files

There are situations where you will need to have zones updated dynamically. Instead of your manually editing a zone file to make changes in a zone, an outside process updates the zone, making changes and saving the file automatically. Dynamic updates are carried out both by master zones updating slave zones and by DHCP servers providing IP addresses they generated for hosts to the DNS server.

A journal file is maintained recording all the changes made to a zone, having a **.jnl** extension. Should a system crash occur, this file is read to implement the most current changes. Should you want to manually update a dynamically updated zone, you will need to erase its journal file first; otherwise, your changes would be overwritten by the journal file entries.

You allow a zone to be automatically updated by specifying the **allow-update** option. This option indicates the host that can perform the update.

```
allow-update {turtle.mytrek.com;};
```

Alternatively, for master zones, you can create a more refined set of access rules using the **update-policy** statement. With the **update-policy** statement, you can list several grant and deny rules for different hosts and types of hosts.

TSIG Signatures and Updates

With BIND 9.*x,* TSIG signature names can be used instead of host names or IP addresses for both **allow-update** and **update-policy** statements (see the following sections on TSIG). Use of TSIG signatures implements an authentication of a host performing a dynamic update, providing a much greater level of security. For example, to allow a DHCP server to update a zone file, you would place an **allow-update** entry in the zone statement listed in the **named.conf** file.

The TSIG key is defined in a key statement, naming the key previously created by the **dnssec-keygen** command. The algorithm is HMAC-MD5, and the secret is the encryption key listed in the **.private** file generated by **dnssec-keygen**.

```
key mydhcpserver {
algorithm HMAC-MD5;
secret "ONQAfbBLnvWU9H8hRqq/WA==";
};
```

The key name can then be used in an **allow-update** or **allow-policy** statement to specify a TSIG key.

```
allow-update { key mydhcpserver;};
```

Manual Updates: nsupdate

You can use the update procedure to perform any kind of update you want. You can perform updates manually or automatically using a script. For DHCP updates, the DHCP server is designed to perform dynamic updates of the DNS server. You will need to configure the DHCP server appropriately, specifying the TSIG key to use and the zones to update.

You can manually perform an update using the **nsupdate** command, specifying the file holding the key with the **-k** option.

```
nsupdate -k myserver.private
```

At the prompt, you can use **nsupdate** commands to implement changes. You match on a record using its full or partial entry. To update a record, you would first delete the old one and then add the changed version, as shown here:

```
update delete rabbit.mytrek.com. A 192.168.0.2
update add rabbit.mytrek.com. A 192.168.0.44
```

DNS Security: Access Control Lists, TSIG, and DNSSEC

DNS security currently allows you to control specific access by hosts to the DNS server, as well as providing encrypted communications between servers and authentication of DNS servers. With access control lists, you can determine who will have access to your DNS server. The DNS Security Extensions (DNSSEC), included with BIND 9.*x,* provide private/public key–encrypted authentication and transmissions. TSIGs (transaction signatures) use shared private keys to provide authentication of servers to secure actions such as dynamic updates between a DNS server and a DHCP server.

Access Control Lists

To control access by other hosts, you use access control lists, implemented with the **acl** statement. **allow** and **deny** options with access control host lists enable you to deny or allow access by specified hosts to the name server. With **allow-query**, you can restrict queries to specified hosts or networks. Normally this will result in a response saying that access is denied. You can further eliminate this response by using the **blackhole** option in the **options** statement.

You define an ACL with the **acl** statement followed by the label you want to give the list and then the list of addresses. Addresses can be IP addresses, network addresses, or a range of

addresses based on CNDR notation. You can also use an ACL as defined earlier. The following example defines an ACL called **mynet**:

```
acl mynet { 192.168.0.1; 192.168.0.2; };
```

If you are specifying a range, such as a network, you also add exceptions to the list by preceding such addresses with an exclamation point (!). In the following example, the **myexceptions** ACL lists all those hosts in the 192.168.0.0 network, except for 192.168.0.3:

```
acl myexceptions {192.168.0.0; !192.168.0.3; };
```

Four default ACLs are already defined for you. You can use them wherever an option uses a list of addresses as an argument. These are **any** for all hosts, **none** for no hosts, **localhost** for all local IP addresses, and **localnet** for all hosts on local networks served by the DNS server.

Once a list is defined, you can then use it with the **allow-query**, **allow-transfer**, **allow-recursion**, and **blackhole** options in a **zone** statement to control access to a zone. **allow-query** specifies hosts that can query the DNS server. **allow-transfer** is used for master/slave zones, designating whether update transfers are allowed. **allow-recursion** specifies those hosts that can perform recursive queries on the server. The **blackhole** option will deny contact from any hosts in its list, without sending a denial response. In the next example, an ACL of mynet is created. Then in the **mytrek.com** zone, only these hosts are allowed to query the server. As the server has no slave DNS serves, zone transfers are disabled entirely. The **blackhole** option denies access from the **myrejects** list, without sending any rejection notice.

```
acl mynet { 192.168.0.0; };
acl myrejects { 10.0.0.44; 10.0.0.93; };

zone "mytrek.com" {
         type master;
         file "mytrek.com";
         allow-query { mynet; };
         allow-recursion { mynet; };
         allow-transfer { none; };
         blackhole {myrejects};
         };
```

Secret Keys

Different security measures will use encryption keys generated with the **dnssec-keygen** command. You can use **dnssec-keygen** to create different types of keys, including zone (ZONE), host (HOST), and user (USER) keys. You specify the type of key with the **-n** option. A zone key will require the name ZONE and the name of the zone's domain name. A zone key is used in DNSSEC operations. The following example creates a zone key for the **mytrek.com** zone:

```
dnssec-keygen -n ZONE mytrek.com.
```

To create a host key, you would use the HOST type. HOST keys are often used in TSIG operations.

```
dnssec-keygen -n HOST turtle.mytrek.com.
```

You can further designate an encryption algorithm (**-a**) and key size (**-b**). Use the **-h** option to obtain a listing of the **dnssec-keygen** options. Currently you can choose from RSA,

DSA, HMAC-MD5, and DH algorithms. The bit range will vary according to the algorithm. RSA ranges from 512 to 4096, and HMAC-MD5 ranges from 1 to 512. The following example creates a zone key using a 768-bit key and the DSA encryption algorithm:

```
dnssec-keygen -a DSA -b 768 -n ZONE mytrek.com.
```

The **dnssec-keygen** command will create public and private keys, each in corresponding files with the suffixes **.private** and **.key**. The **.key** file is a KEY resource record holding the public key. For DNSSEC, the private key is used to generate signatures for the zone, and the public key is used to verify the signatures. For TSIG, a shared private key generated by the HMAC-MD5 algorithm is used instead of a public/private key pair.

DNSSEC

DNSSEC provides encrypted authentication to DNS. With DNSSEC, you can create a signed zone that is securely identified with an encrypted signature. This form of security is used primarily to secure the connections between master and slave DNS servers, so that a master server transfers update records only to authorized slave servers and does so with a secure encrypted communication. Two servers that establish such a secure connection do so using a pair of public and private keys. In effect, you have a parent zone that can securely authenticate child zones, using encrypted transmissions. This involves creating zone keys for each child and having those keys used by the parent zone to authenticate the child zones.

Zone Keys

You generate a zone key using the **dnssec-keygen** command and specifying the zone type, ZONE, with the **-n** option. For the key name, you use the zone's domain name. The following example creates a zone key for the **mytrek.com** zone:

```
dnssec-keygen -n ZONE mytrek.com.
```

You can further designate an encryption algorithm (**-a**) and a key size (**-b**). Use the **-h** option to obtain a listing of the **dnssec-keygen** options. Since you are setting up a public/private key pair, you should choose either the RSA or DSA algorithm. The bit range will vary according to the algorithm. RSA ranges from 512 to 4096, and DSA ranges from 512 to 1024. The following example creates a zone key using a 768-bit key and the DSA encryption algorithm:

```
dnssec-keygen -a DSA -b 768 -n ZONE mytrek.com.
```

The **dnssec-keygen** command will create public and private keys, each in corresponding files with the suffixes **.private** and **.key**. The private key is used to generate signatures for the zone, and the public key is used to verify the signatures. The **.key** file is a KEY resource record holding the public key. This is used to decrypt signatures generated by the corresponding private key. You add the public key to the DNS configuration file, **named.conf**, using the **$INCLUDE** statement to include the **.key** file.

DNSSEC Resource Records

In the zone file, you then use three DNSSEC DNS resource records to implement secure communications for a given zone: KEY, SIG, and NXT. In these records, you use the signed keys for the zones you have already generated. The KEY record holds public keys associated with zones, hosts, or users. The SIG record stores digital signatures and expiration dates for a set of resource

records. The NXT record is used to determine that a resource record for a domain does not exist. In addition, several utilities let you manage DNS encryption. With the **dnskeygen** utility, you generated the public and private keys used for encryption. **dnssigner** signs a zone using the zone's private key, setting up authentication.

```
mytrek.com. KEY 0x4101 3 3 (
AvqyXgKk/uguxkJF/hbRpYzxZFG3x8EfNX389l7GX6w7rlLy
BJ14TqvrDvXr84XsShg+OFcUJafNr84U4ER2dg6NrlRAmZA1
jFfV0UpWDWcHBR2jJnvgV9zJB2ULMGJheDHeyztM1KGd2oGk
Aensm74NlfUqKzy/3KZ9KnQmEpj/EEBr48vAsgAT9kMjN+V3
NgAwfoqgS0dwj5OiRJoIR4+cdRt+s32OUKsclAODFZTdtxRn
vXF3qYV0S8oewMbEwh3trXi1c7nDMQC3RmoY8RVGt5U6LMAQ
KITDyHU3VmRJ36vn77QqSzbeUPz8zEnbpik8kHPykJZFkcyj
jZoHT1xkJ1tk )
```

To secure a DNS zone with DNSSEC, you first use **dnskeygen** to create public and private keys for the DNS zone. Then use **dnssigner** to create an authentication key. In the DNS zone file, you enter a KEY resource record in which you include the public key. The public key will appear as a lengthy string of random characters. For the KEY record, you enter in the domain name followed by the KEY and then the public key.

For authentication, you can sign particular resource records for a given domain or host. Enter the domain or host followed by the term **SIG** and then the resource record's signature.

```
mytrek.com. SIG KEY 3 86400 19990321010705 19990218010705 4932 com. (
Am3tWJzEDzfU1xwg7hzkiJ0+8UQaPtlJhUpQx1snKpDUqZxm
igMZEVk= )
```

The NXT record lets you negatively answer queries.

```
mytrek.com. NXT ftp.mytrek.com. A NS SOA MX SIG KEY NXT
```

Signing Keys

To set up secure communications between a parent (master) DNS server and a child (slave) DNS server, the public key then needs to be sent to the parent zone. There, the key can be signed by the parent. As you may have more than zone key, you create a keyset using the **dnssec-makekeyset** command. This generates a file with the extension **.keyset** that is then sent to the parent. The parent zone then uses the **dnssec-signkey** command to sign a child's keyset. This generates a file with the prefix **signedkey-**. This is sent back to the child and now contains both the child's keyset and the parent's signatures. Once the child has the **signedkey-** files, the **dnssec-signedzone** command can be used to sign the zone. The **dnssec-signedzone** command will generate a file with the extension **.signed**. This file is then included in the **named.conf** file with the INCLUDE operation. The **trusted-keys** statement needs to list the public key for the parent zone.

TSIG Keys

TSIG (transaction signatures) also provide secure DNS communications, but they share the private key instead of a private/public key pair. They are usually used for communications between two local DNS servers, and to provide authentication for dynamic updates such as those between a DNS server and a DHCP server.

Note: For BIND 8.0, you use **dnskeygen** instead of **dnssec-keygen**.

Generating TSIG keys

To create a TSIG key for your DNS server, you use the **dnssec-keygen** command as described earlier. Instead of using the same keys you use for DNSSEC, you create a new set to use for transaction signatures. For TSIG, a shared private key is used instead of a public/private key pair. For a TSIG key you would use an HMAC-MD5 algorithm that generates the same key in the both the **.key** and **.private** files. Use the **-a** option to specify the HMAC-MD5 algorithm to use and the **-b** option for the bit size. (HMAC-MD5 ranges from 1 to 512.) Use the **-n** option to specify the key type, in this case HOST for the host name. The bit range will vary according to the algorithm. The following example creates a host key using a 128-bit key and the HMAC-MD5 encryption algorithm:

```
dnssec-keygen -a HMAC-MD5 -b 128 -n HOST turtle.mytrek.com
```

This creates a private key and a public key, located in the **.key** and **.private** files. In a TSIG scheme, both hosts would use the same private key for authentication. For example, to enable a DHCP server to update a DNS server, both would need the private (secret) key for a TSIG authentication. The HMAC-MD5 key is used as a shared private key, generating both the same private and public keys in the **.key** and **.private** files.

The Key Statement

You then specify a key in the **named.conf** file with the **key** statement. For the algorithm option, you list the HMAC-MD5 algorithm, and for the secret option, you list the private key. This key will be listed in both the **.private** and **.key** files. The preceding example would generate key and private files called **Kturtle.mytrek.com.+157.43080.key** and **Kturtle.mytrek.com.+157.43080.private**. The contents of the **.key** file consist of a resource record shown here:

```
turtle.mytrek.com. IN KEY 512 3 157 ONQAfbBLnvWU9H8hRqq/WA==
```

The contents of the private file show the same key along with the algorithm:

```
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: ONQAfbBLnvWU9H8hRqq/WA==
```

Within the **named.conf** file, you then name the key using a **key** statement:

```
key myserver {
algorithm HMAC-MD5;
secret "ONQAfbBLnvWU9H8hRqq/WA==";
};
```

The key's name can then be used to reference the key in other named statements, such as **allow-update** statements:

```
allow-update myserver;
```

The DNS server or DHCP server with which you are setting up communication will also have to have the same key. See the earlier section "Dynamic Update: DHCP and Journal Files". For communication between two DNS servers, each would have to have a server statement specifying

the shared key. In the following example, the **named.conf** file for the DNS server on 192.168.0.1 would have to have the following server statement to communicate with the DNS server on 10.0.0.1, using the shared myserver key. The **named.conf** file on the 10.0.0.1 DNS server would have to have a corresponding server statement for the 192.168.0.1 server.

```
server 10.0.0.1 {  keys (myserver;}; };
```

Split DNS: Views

BIND 9.*x* allows you to divide DNS space into internal and external views. This organization into separate views is referred to as split DNS. Such a configuration is helpful to manage a local network that is connected to a larger network, such as the Internet. Your internal view would include DNS information on hosts in the local network, whereas an external view would show only the part of the DNS space that is accessible to other networks. DNS views are often used when you have a local network that you want to protect from a larger network such as the Internet. In effect, you protect DNS information for hosts on a local network from a larger external network such as the Internet.

Internal and External Views

The internal DNS servers will hold DNS information about local hosts. The external DNS view maintains connections to the Internet through a gateway as well as manages DNS information about any local hosts that allow external access, such as FTP or Web sites. The gateways and Internet-accessible sites make up the external view of hosts on the network. The internal view handles all queries to the local hosts or subdomains. Queries to external hosts such as Internet sites are sent to the external servers, which then forward them on to the Internet. Queries sent to those local hosts that operate external servers such as Internet FTP and Web sites are sent to the external DNS view for processing. Mail sent to local hosts from the Internet is handled first by the external servers, which then forward messages on to the internal servers. With a split DNS configuration, local hosts can access other local hosts, Internet sites, and local hosts maintaining Internet servers. Internet users, on the other hand, can access only those hosts open to the Internet (served by external view) such as those with Internet servers like FTP and HTTP. Internet users can, however, send mail messages to any of the local hosts, internal and external.

You can also use DNS views to manage connections between a private network that may use only one Internet address to connect its hosts to the Internet. In this case, the internal view holds the private addresses (192.168...), and the external view connects a gateway host with an Internet address to the Internet. This adds another level of security, providing a result similar to IP masquerading.

Configuring Views for separate servers

DNS views are configured with the allow statements such as **allow-query** and **allow-transfer**. With these statements, you can specify the hosts that a zone can send and receive queries and transfers to and from. For example, the internal zone could accept queries from other local hosts, but not from local hosts with external access such as Internet servers. The local Internet servers, though, can accept queries from the local hosts. All Internet queries are forwarded to the gateway. In the external configuration, the local Internet servers can accept queries from anywhere. The gateways receive queries from both the local hosts and the local Internet servers.

In the following examples, a network of three internal hosts and one external host is set up into a split view. There are two DNS servers: one for the internal network and one for external access, based on the external host. In reality these make up one network but they are split into two views. The internal view is known as **mygolf.com**, and the external as **greatgolf.com**. In each configuration, the internal hosts are designated in ACL-labeled internals, and the external host is designated in ACL-labeled externals. Should you want to designate an entire IP address range as internal, you could simply use the network address, as in 192.168.0.0/24. In the options section, **`allow-query`**, **`allow-recursion`**, and **`allow-transfers`** restrict access within the network.

To implement a split DNS space, you can either set up different DNS servers for the internal and external views, or use the view statement for a single DNS server.

The following example shows only the configuration entries needed to implement an internal view. In the **mygolf.com** zone, queries and transfers are allowed only among internal hosts. The global **`allow-recursion`** option allows recursion among internals.

Internal DNS server

```
acl internals { 192.168.0.1; 192.168.0.2; 192.168.0.3; };
acl externals {10.0.0.1;};

options {
         forward only;
         forwarders {10.0.0.1;}; // forward to external servers
         allow-transfer { none; }; // allow-transfer to no one by default
         allow-query { internals; externals; };// restrict query access
         allow-recursion { internals; }; // restrict recursion to internals
         }


include "/etc/named.rfc1912.zones

zone "mygolf.com" {
              type master;
              file "mygolf.db";
              forwarders { };
              allow-query { internals; };
              allow-transfer { internals; }
              };
```

In the configuration for the external DNS server, the same ACLs are set up for internals and externals. In the **`options`** statement, recursion is now allowed for both externals and internals. In the **mygolf.com** zone, queries are allowed from anywhere, and recursion is allowed for externals and internals. Transfers are not allowed at all.

External DNS server

```
acl internals { 192.168.0.1; 192.168.0.2; 192.168.0.3; };
acl externals {10.0.0.1;};

options {
         allow-transfer { none; }; // allow-transfer to no one
         allow-query { internals; externals; };// restrict query access
         allow-recursion { internals; externals }; // restrict recursion
         };


zone "greatgolf.com" {
               type master;
               file "greatgolf.db";
               allow-query { any; };
               allow-transfer { internals; externals; };
            };
```

Split View on a single DNS server using the view statements

Instead of using two separate physical servers, you can use the view statement to configure split views on a single DNS server. If you use view statements, then all zone definitions have to be included in a view statement.

The following is based on the DNS sample in the **/usr/share/doc/BIND*/sample/etc** directory for a **named.conf** file that uses views to define internal and external servers. Check this example for a detailed description of a views implementation. The internal server are labeled "internal" and the external servers as "external." For a caching only server you could use a view with the label "localhost_resolver", as used in the Fedora BIND example.

You use the **match-clients** option to specify allowable clients. The **recursion** option can allow recursion access, allowing the DNS server to submit queries to other DNS servers it does not control. The internal view loads the **named.rfc1912.zones** file for localhost access.

The external view allows access from any host, but does turns **recursion** off, responding only to queries to itself and the DNS servers it controls.

named.conf for split views

```
options
      {
       directory "/var/named"; // the default
       dump-file             "/var/named/data/cache_dump.db";
        statistics-file      "/var/named/data/named_stats.txt";
        memstatistics-file   "/var/named/data/named_mem_stats.txt";

      };


acl internals { 192.168.0.1; 192.168.0.2; 192.168.0.3; };
acl externals {10.0.0.1;};


view  "internal"
       {
        match-clients {localhost; internals;};
       recursion yes;
        include "/etc/named.root.hints";
        include "/etc/named.rfc1912.zones";


        zone "mygolf.com" {
              type master;
              file "mygolf.db";
              allow-transfer { internals; }
         };
     };


view   "external"
       {
        match-clients { any; };

       recursion no;
       include "/etc/named.root.hints";

        zone "greatgolf.com" {
              type master;
              file "greatgolf.db";
              allow-transfer { internals; externals; };
              }
       };
```

fedora

31. Network Auto-configuration with IPv6, DHCPv6, and DHCP

Many networks now provide either IPv6 autoconfiguration or the DHCP (Dynamic Host Configuration Protocol) service, which automatically provides network configuration for all connected hosts. Autoconfiguration can be either stateless, as in the case of IPv6, or stateful, as with DHCP. Stateless IPv6 autoconfiguration requires no independent server or source to connect to a network. It is a direct plug-and-play operation, where the hardware network interfaces and routers can directly determine the correct addresses. DCHP is an older method that requires a separate server to manage and assign all addresses. Should this server ever fail, hosts cannot connect.

With the DHCP protocol, an administrator uses a pool of IP addresses from which the administrator can assign an IP address to a host as needed. The protocol can also be used to provide all necessary network connection information such as the gateway address for the network or the netmask. Instead of having to configure each host separately, network configuration can be handled by a central DHCP server. The length of time that an address can be used can be controlled by means of leases, making effective use of available addresses. If your network is configuring your systems with DHCP, you will not have to configure it.

There are currently two versions of DHCP, one for the original IPv4 protocol and another, known as DHCPv6, for the IPv6 protocol. The IPv6 protocol includes information for dynamic configuration that the IPv4 protocol lacks. In this respect, the IPv4 protocol is much more dependent on DHCP than IPv6 is.

IPv6 Stateless Autoconfiguration

In an IPv6 network, the IPv6 protocol includes information that can directly configure a host. With IPv4 you either had to manually configure each host or rely on a DHCP server to provide configuration information. With IPv6, configuration information is integrated into the Internet protocol directly. IPv6 address autoconfiguration is described in detail in RFC 2462.

IPv6 autoconfiguration capabilities are known as stateless, meaning that it can directly configure a host without recourse of an external server. Alternatively, DHCP, including DHCPv6, is stateful, where the host relies on an external DHCP server to provide configuration information. Stateless autoconfiguration has the advantage of hosts not having to rely on a DHCP server to maintain connections to a network. Networks can even become mobile, hooking into one subnet or another, automatically generating addresses as needed. Hosts are no longer tied to a particular DHCP server.

Generating the Local Address

To autoconfigure hosts on a local network, IPv6 makes use of the each network device's hardware MAC address. This address is used to generate a temporary address, with which the host can be queried and configured.

The MAC address is used to create a link-local address, one with a link-local prefix, **FE80::0**, followed by an interface identifier. The link-local prefix is used for physically connected hosts such as those on a small local network.

A uniqueness test is then performed on the generated address. Using the Neighbor Discovery Protocol (NDP), other hosts on the network are checked to see if another host is already using the generated link-local address. If no other host is using the address, then the address is

assigned for that local network. At this point the host has only a local address valid within the local physical network. Link-local addresses cannot be routed to a larger network.

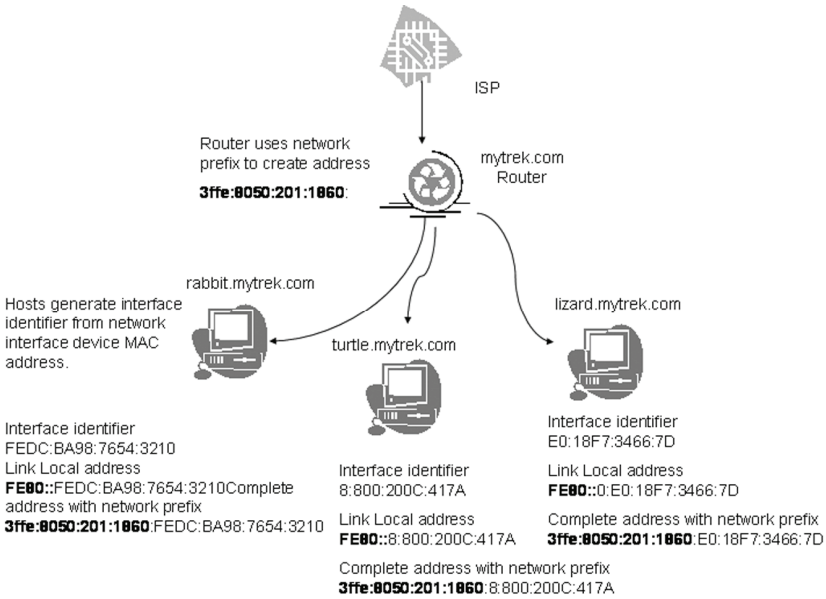


Figure 31-1: Stateless IPv6 address autoconfiguration

Generating the Full Address: Router Advertisements

Once the link-local address has been determined, the router for the network is queried for additional configuration information. The information can be either stateful or stateless, or both. For stateless configuration, information such as the network address is provided directly, whereas for stateful configuration, the host is referred to a DHCPv6 server where it can obtain configuration information. The two can work together. Often the stateless method is used for addresses, and the stateful DHCPv6 server is used to provide other configuration information such as DNS server addresses.

In the case of stateless addresses, the router provides the larger network address, such as the network's Internet address. This address is then added to the local address, replacing the original link-local prefix, giving either a complete global Internet address or, in the case of private networks, unique-local addresses. Routers will routinely advertise this address information, though it can also be specifically requested. The NDP is used to query the information. Before the address is officially assigned, a duplicate address detection procedure checks to see if the address is already in use. The process depends on the router's providing the appropriate addressing information in the form of router advertisements. If there is no router, or there are no route advertisements, then a stateful method like DHCPv6 or manual configuration must be used to provide the addresses.

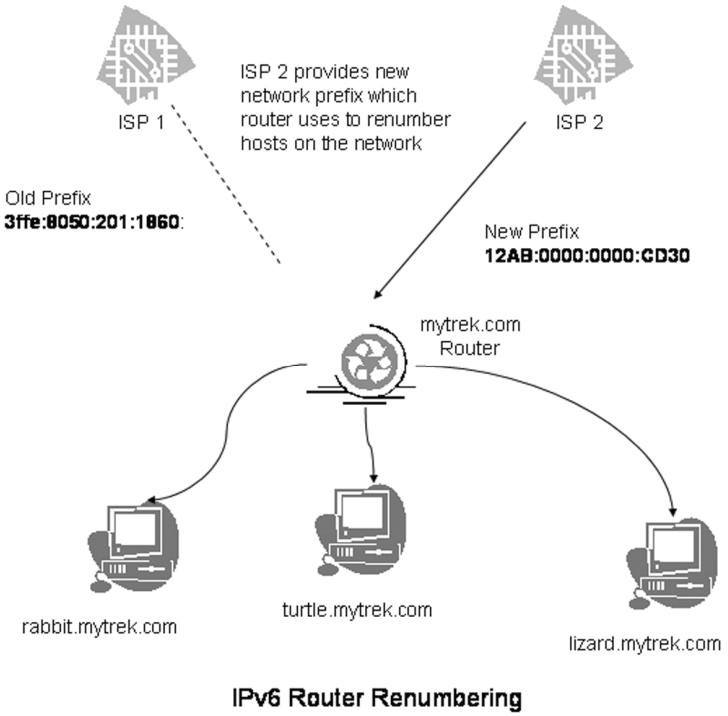


Figure 31-2: Router renumbering with IPv6 autoconfiguration

Figure 31-1 shows a network that is configured with stateless address autoconfiguration. Each host first determines its interface identifier using its own MAC hardware address to create a temporary link-local address for each host using the **FE80::0** prefix. This allows initial communication with the network's router. The router then uses its network prefix to create full Internet addresses, replacing the link-local prefix.

Router Renumbering

With IPv6, routers have the ability to renumber the addresses on their networks by changing the network prefix. Renumbering is carried out through the Router Renumbering (RR) Protocol. (See RFC 2894 for a description of router renumbering.) Renumbering is often used when a network changes ISP providers and requires that the net address for all hosts be changed (see Figure 31-2). It can also be used for mobile networks where a network can be plugged in to different larger networks, renumbering each time.

With renumbering, routers place a time limit on addresses, similar to the lease time in DHCP, by specifying an expiration limit for the network prefix when the address is generated. To ease transition, interfaces still keep their old addresses as deprecated addresses, while the new one are first being used. The new ones will be the preferred addresses used for any new connections,

while deprecated ones are used for older connections. In effect, a host can have two addresses, one deprecated and one preferred. This regeneration of addresses effectively renumbers the hosts.

IPv6 Stateful Autoconfiguration: DHCPv6

The IPv6 version of DHCP (DHCPv6) provides stateful autoconfiguration to those networks that still want a DHCP-like service on IPv6 networks. DHCP IPv6 provides configuration information from a server, just like DHCP, but it is a completely different protocol from the IPv4 version, with different options and capabilities. As a stateful configuration process, information is provided by an independent server. A DHCP IPv6 server and client are available in the Fedora repository. You can file out more about the DHCPv6 project at **https://fedorahosted.org/dhcpv6/**. The server requires its own DHCPv6 clients.

DHCPv6 uses a its own set of options for both the client and the server. The DHCP server for IPv6 is called **dhcp6s**, and the DHCP client for IPv6 is **dhcp6c**. Their corresponding configuration files are **/etc/dhcp6c** and **/etc/sysconfig/dhcp6s**. A service script, **/etc/init.d/dhcp6s**, can be used to manage the dhcp6s server.

As with IPv6 autoconfiguration, the host identifier for a local address is first automatically generated. This is a local-link address containing a host identifier address generated from the host interface's MAC address.

Once the local-link address is determined, the router is queried for the DHCPv6 server. This information is provided in router advertisements that are broadcast regularly. At this point the two different kinds of stateful information can be provided by the server: addresses and other configuration information. The host is notified which kinds of stateful information are provided. If address information is not given by the DHCPv6 server, then addresses will be determined using the stateless autoconfiguration method described in the preceding section. If address information is provided, then an address will be obtained from the server instead of being directly generated. Before leasing an address, the server will run a duplicate address detection procedure to make sure the address is unique.

The dhcpv6 package intalls the **dhcpv6s** server. Its configuration file is **/etc/dhcpv6s.conf** and its start up script is **/etc/init.d/dhspv6s**.

```
service start hdspv6s
```

Linux as an IPv6 Router: radvd

For a Linux system that operates as a router, you use the **radvd** (Router ADVertisement Daemon) to advertise addresses, specifying a network prefix in the **/etc/radvd.conf** file (Fedora repository). The **radvd** daemon will detect router network address requests from hosts, known as router solicitations, and provide them with a network address using a router advertisement. These router advertisements will also be broadcast to provide the network address to any hosts that do not send in requests. For **radvd** to work, you will have to turn on IPv6 forwarding. Use **sysctl** and set **net.ipv6.conf.all.forwarding** to 1. To start up the **radvd** daemon, you use the **`radvd`** startup script. To check the router addresses **radvd** is sending, you can use **`radvdump`**.

You will have to configure the **radvd** daemon yourself, specifying the network address to broadcast. Configuration, though, is very simple, as the full address will be automatically generated using the host's hardware address. A configuration consists of interface entries, which in turn lists

interface options, prefix definitions, and options, along with router definitions if needed. The configuration is placed in the **/etc/radvd.conf** file, which will look something like this:

```
interface eth0 {
     AdvSendAdvert on;
          prefix fc00:0:0:0::/64
          {
          AdvOnLink on;
          AdvAutonomous on;
          };
};
```

This assumes one interface is used for the local network, **eth0**. This interface configuration lists an interface option (AdvSendAdvert) and a prefix definition, along with two prefix options (AdvOnLink and AdvAutonomous). To specify prefix options for a specific prefix, add them within parentheses following the prefix definition. The prefix definition specifies your IPv6 network address. If a local area network has its own network address, you will need to provide its IPv6 network prefix address. For a private network, such as a home network, you can use the unique-local IPv6 prefix, which operates like the IPv4 private network addresses, 192.168.0. The preceding example uses a unique-local address that is used for private IPv6 networks, fc00:0:0:0::, which has a length of 64 bits.

The AdvSendAdvert interface option turns on network address advertising to the hosts. The AdvAutonomous network prefix option provides automatic address configuration, and AdvOnLink simply means that host requests can be received on the specified network interface.

A second network interface is then used to connect the Linux system to an ISP or larger network. If the ISP supports IPv6, this is simply a matter of sending a router solicitation to the ISP router. This automatically generates your Internet address using the hardware address of the network interface that connects to the Internet and the ISP router's advertised network address. In Figure 31-2, shown earlier, the **eth0** network interface connects to the local network, whereas **eth1** connects to the Internet.

DHCPv6 as a client only: dhcp6c

If you are not running a DHCP server, and are only using your system as a client for your network's DHCP server running a DHCPv6 server, then you will find that Fedora will install the DHCPv6 client software for you. Client support is carried out by the **dhcp6c** tool. When your network starts up, it uses **dhcp6c** to set up your DHCP connection. Though defaults are usually adequate, you can further configure the DHCP client using the **/etc/dhcp6c.conf** file. Consult the **dhcp6c.conf** Man page for a detailed list of configuration options. You can also directly run **dhcp6c**.

```
dhcp6c
```

DHCP for IPv4

DHCP provides configuration information to systems connected to a IPv4 TCP/IP network, whether the Internet or an intranet. The machines on the network operate as DHCP clients, obtaining their network configuration information from a DHCP server on their network. A

machine on the network runs a DHCP client daemon that automatically receives its network configuration information from its network's DHCP server. The information includes its IP address, along with the network's name server, gateway, and proxy addresses, including the netmask. Nothing has to be configured manually on the local system, except to specify the DHCP server it should get its network configuration from. This has the added advantage of centralizing control over network configuration for the different systems on the network. A network administrator can manage the network configurations for all the systems on the network from the DHCP server.

A DHCP server also supports several methods for IP address allocation: automatic, dynamic, and manual. Automatic allocation assigns a permanent IP address for a host. Manual allocation assigns an IP address designated by the network administrator. With dynamic allocation, a DHCP server can allocate an IP address to a host on the network only when the host actually needs to use it. Dynamic allocation takes addresses from a pool of IP addresses that hosts can use when needed and release when they are finished.

The current version of DHCP now supports the DHCP failover protocol, in which two DHCP servers support the same address pool. Should one fail, the other can continue to provide DHCP services for a network. Both servers are in sync and have the same copy of network support information for each host on the network. Primary and secondary servers in this scheme are designated with the primary and secondary statements.

A variety of DHCP servers and clients are available for different operating systems. The Fedora repository provides DHCP version 3 software from the Internet Software Consortium (ISC) at **www.isc.org**. The software available includes a DHCP server, a client, and a relay agent. The DHCP client is called **dhclient**, and the IPv4 server is called **dhcpd**.

Configuring DHCP IPv4 Client Hosts

Configuring hosts to use a DHCP server is a simple matter of setting options for the host's network interface device, such as an Ethernet card. For a Linux host, you can use a distribution network tool to set the host to automatically access a DHCP server for network information. On a network tool's tab for configuring the Internet connection, you will normally find a check box for selecting DHCP. Clicking this box will enable DHCP.

Client support is carried out by the **dhclient** tool. When your network starts up, it uses **dhclient** to set up your DHCP connection. Though defaults are usually adequate, you can further configure the DHCP client using the **/etc/dhclient.conf** file. Consult the **dhclient.conf** Man page for a detailed list of configuration options. **dhclient** keeps lease information on the DCHP connection in the **/var/lib/dhcp/dhclient.leases** file. You can also directly run **dhclient** to configure DHCP connections. Keep in mind that dhclient is the DHCPv4 version, not the DHCPv6 version (**dhcp6c**).

```
dhclient
```

Configuring the DHCP IPv4 Server

You can stop and start the DHCP server using the **dhcpd** script in the **/etc/rc.d/init.d** directory. Use the services-admin tool or the **dhcpd** script with the **start**, **restart**, and **stop** options. The following example starts the DHCP server. Use the **stop** option to shut it down and **restart** to restart it.

```
sevice dhcpd start
```

Dynamically allocated IP addresses, known as *leases,* will be assigned for a given time. When a lease expires, it can be extended or a new one generated. Current leases are listed in the **dhcpd.leases** file located in the **/var/lib/dhcp** directory. A lease entry will specify the IP address and the start and end times of the lease along with the client's hostname.

GNOME DHCPD Configuration, GDHCPD

You can also use the GNOME DHCPD configuration tool (**GDHCPD** package, to configure DHCP server graphically. Though still in early development, the GDHCPD too provides an easy to use method of setting up your server. The package will be installed on the Administration | System Tools menu. You may need to use gsku to start it.

```
gksu gdhcpd
```

On the Scopes tab you enter the network connection, IP address for that connection and the netmask. Then click Add. In the Range from boxes, enter a range of addresses to allocate, then click Add.

The single hosts tab lets you specify static IP addresses to assign to particular hosts. These are normally used for systems like servers whose IP addresses do not usually change.

Click the Settings button to see your server configuration settings like the configuration and leases files used, and whether to allow DNS updates.

/etc/dhcpd.conf

The configuration file for the DHCP server is **/etc/dhcpd.conf**, where you specify parameters and declarations that define how different DHCP clients on your network are accessed by the DHCP server, along with options that define information passed to the clients by the DHCP server. These parameters, declarations, and options can be defined globally for certain subnetworks or for specific hosts. Global parameters, declarations, and options apply to all clients, unless overridden by corresponding declarations and options in subnet or host declarations. Technically, all entries in a **dhcpd.conf** file are statements that can be either declarations or parameters. All statements end with a semicolon. Options are specified in **`options`** parameter statements. Parameters differ from declarations in that they define if and how to perform tasks, such as how long a lease is allocated. Declarations describe network features such as the range of addresses to allocate or the networks that are accessible. See Table 31-1 for a listing of commonly used declarations and options.

Declarations provide information for the DHCP server or designate actions it is to perform. For example, the **`range`** declaration is used to specify the range of IP addresses to be dynamically allocated to hosts:

```
range 192.168.0.5 192.168.0.128;
```

With parameters, you can specify how the server is to treat clients. For example, the **`default-lease-time`** declaration sets the number of seconds a lease is assigned to a client. The **`filename`** declaration specifies the boot file to be used by the client. The **`server-name`** declaration informs the client of the host from which it is booting. The **`fixed-address`** declaration can be used to assign a static IP address to a client. See the Man page for **dhcpd.conf** for a complete listing.

Options provide information to clients that they may need to access network services, such as the domain name of the network, the domain name servers that clients use, or the broadcast address. See the Man page for **dhcp-options** for a complete listing. This information is provided by **option** parameters as shown here:

```
option broadcast-address 192.168.0.255;
option domain-name-servers 192.168.0.1, 192.168.0.4;
option domain-name "mytrek.com";
```

Your **dhcpd.conf** file will usually begin with declarations, parameters, and options that you define for your network serviced by the DHCP server. The following example provides router (gateway), netmask, domain name, and DNS server information to clients. Additional parameters define the default and maximum lease times for dynamically allocated IP addresses.

```
option routers 192.168.0.1;
option subnet-mask 255.255.255.0;
option domain-name "mytrek.com ";
option domain-name-servers 192.168.0.1;
default-lease-time 21600;
max-lease-time 43200;
```

With the subnet, host, and group declarations, you can reference clients in a specific network, particular clients, or different groupings of clients across networks. Within these declarations, you can enter parameters, declarations, or options that will apply only to those clients. Scoped declarations, parameters, and options are enclosed in braces. For example, to define a declaration for a particular host, you use the **host** declaration as shown here:

```
host rabbit {
        declarations, parameters, or options;
        }
```

You can collect different subnet, global, and host declaration into groups using the **group** declaration. In this case, the global declarations are applied only to those subnets and hosts declared within the group.

Dynamic IPv4 Addresses for DHCP

Your DHCP server can be configured to select IP addresses from a given range and assign them to different clients. Given a situation where you have many clients that may not always be connected to the network, you can effectively service them with a smaller pool of IP addresses. IP addresses are assigned only when they are needed. With the **range** declaration, you specify a range of addresses that can be dynamically allocated to clients. The declaration takes two arguments, the first and last addresses in the range.

```
range 192.168.1.5 192.168.1.128;
```

For example, if you are setting up your own small home network, you would use a network address beginning with 192.168. The range would specify possible IP addresses with that network. So, for a network with the address 192.168.0.0, you place a **range** declaration along with any other information you want to give to your client hosts. In the following example, a range of IP addresses extending from 192.168.0.1 to 192.168.0.128 can be allocated to the hosts on that network:

```
range 192.168.0.5 192.168.0.128;
```

Entries	Description
Declarations	
shared-network *name*	Indicates if some subnets share the same physical network.
subnet *subnet-number netmask*	References an entire subnet of addresses.
range [*dynamic-bootp*] *low-address* [*high-address*];	Provides the highest and lowest dynamically allocated IP addresses.
host *hostname*	References a particular host.
group	Lets you label a group of parameters and declarations and then use the label to apply them to subnets and hosts.
allow unknown-clients; **deny unknown-clients;**	Does not dynamically assign addresses to unknown clients.
allow bootp; deny bootp;	Determines whether to respond to **bootp** queries.
allow booting; deny booting;	Determines whether to respond to client queries.
Parameters	
default-lease-time *time;*	Assigns length in seconds to a lease.
max-lease-time *time;*	Assigns maximum length of lease.
hardware *hardware-type hardware-address;*	Specifies network hardware type (Ethernet or token ring) and address.
filename "*filename*"**;**	Specifies name of the initial boot file.
server-name "*name*"**;**	Specifies name of the server from which a client is booting.
next-server *server-name;*	Specifies server that loads the initial boot file specified in the filename.
fixed-address *address* [, *address ...*]**;**	Assigns a fixed address to a client.
get-lease-hostnames *flag;*	Determines whether to look up and use IP addresses of clients.
authoritative; **not authoritative;**	Denies invalid address requests.
server-identifier hostname;	Specifies the server.
Options	
option subnet-mask *ip-address;*	Specifies client's subnet mask.
option routers *ip-address* [, *ip-address...*]**;**	Specifies list of router IP addresses

	on client's subnet.
option domain-name-servers *ip-address* **[**, *ip-address...* **];**	Specifies list of domain name servers used by the client.
option log-servers *ip-address* **[**, *ip-address...* **];**	Specifies list of log servers used by the client.
option host-name *string;*	Specifies client's hostname.
option domain-name *string;*	Specifies client's domain name.
option broadcast-address *ip-address;*	Specifies client's broadcast address.
option nis-domain *string;*	Specifies client's Network Information Service domain.
option nis-servers *ip-address* **[**, *ip-address...* **];**	Specifies NIS servers the client can use.
option smtp-server *ip-address* **[**, *ip-address...* **];**	Lists SMTP servers used by the client.
option pop-server *ip-address* **[**, *ip-address...* **];**	Lists POP servers used by the client.
option nntp-server *ip-address* **[**, *ip-address...* **];**	Lists NNTP servers used by the client.
option www-server *ip-address* **[**, *ip-address...* **];**	Lists web servers used by the client.

Table 31-1: DHCP Declarations, Parameters, and Options

You should also define your lease times, both a default and a maximum:

```
default-lease-time 21600;
max-lease-time 43200;
```

For a small, simple home network, you just need to list the **range** declaration along with any global options as shown here. If your DHCP server is managing several subnetworks, you will have to use the **subnet** declarations.

In order to assign dynamic addresses to a network, the DHCP server will require that your network topology be mapped. This means it needs to know what network addresses belong to a given network. Even if you use only one network, you will need to specify the address space for it. You define a network with the **subnet** declaration. Within this **subnet** declaration, you can specify any parameters, declarations, or options to use for that network. The **subnet** declaration informs the DHCP server of the possible IP addresses encompassed by a given subnet. This is determined by the network IP address and the netmask for that network. The next example defines a local network with address spaces from 192.168.0.0 to 192.168.0.255. The **range** declaration allows addresses to be allocated from 192.168.0.5 to 192.168.0.128.

```
subnet 192.168.1.0 netmask 255.255.255.0 {
          range 192.168.0.5 192.168.0.128;
 }
```

Versions of DHCP prior to 3.0 required that you even map connected network interfaces that are not being served by DHCP. Thus each network interface has to have a corresponding **subnet** declaration. Those not being serviced by DHCP don't have a **not authoritative** parameter as shown here (192.168.2.0 being a network not to be serviced by DHCP). In version 3.0 and later, DHCP simply ignores unmapped network interfaces:

```
subnet 192.168.2.0 netmask 255.255.255.0 {
      not authoritative;
}
```

The implementation of a very simple DHCP server for dynamic addresses is shown in the sample **dhcpd.conf** file that follows:

/etc/dhcp3/dhcpd.conf

```
option routers 192.168.0.1;
 option subnet-mask 255.255.255.0;
 option domain-name "mytrek.com ";
 option domain-name-servers 192.168.0.1;

subnet 192.168.1.0 netmask 255.255.255.0 {
         range 192.168.0.5 192.168.0.128;
         default-lease-time 21600;
         max-lease-time 43200;
         }
```

DHCP Dynamic DNS Updates

For networks that also support a Domain Name Server, dynamic allocation of IP addresses currently needs to address one major constraint: DHCP needs to sync with a DNS server. A DNS server associates hostnames with particular IP addresses, whereas in the case of dynamic allocation, the DHCP server randomly assigns its own IP addresses to different hosts. These may or may not be the same as the IP addresses that the DNS server expects to associate with a hostname. A solution to this problem is being developed, called Dynamic DNS. With Dynamic DNS, the DHCP server is able to automatically update the DNS server with the IP addresses the DHCP server has assigned to different hosts.

Note: Alternatively, if you want to statically synchronize your DHCP and DNS servers with fixed addresses, you configure DHCP to assign those fixed addresses to hosts. You can then have the DHCP server perform a DNS lookup to obtain the IP address it should assign, or you can manually assign the same IP address in the DHCP configuration file. Performing a DNS lookup has the advantage of specifying the IP address in one place, the DNS server.

The DHCP server has the ability to dynamically update BIND DNS server zone configuration files. You enable dynamic updates on a DNS server for a zone file by specifying the **allow-update** option for it in the **named.conf** file. Furthermore, it is strongly encouraged that you use TSIG signature keys to reference and authenticate the BIND and DHCP servers. Currently, DHCP uses the Interim DNS Update Scheme to perform dynamic DNS updates, replacing an earlier Ad-Hoc DNS Update Scheme. A finalized version will be implemented in future DHCP releases. You can find detailed information about dynamic DNS in the **dhcpd.conf** Man page.

Enabling the use of a TSIG key involves syncing configurations for both your DHCP and DNS servers. Both have to be configured to use the same key for the same domains. First you need to create a shared secret TSIG signature key using **dnssec-keygen**,. In the DNS server, you place TSIG key declarations and **allow-update** entries in the server's **named.conf** file, as shown in this example:

```
key mydhcpserver {
algorithm HMAC-MD5;
secret "ONQAfbBLnvWU9H8hRqq/WA==";
};


zone "mytrek.com" {
       type master;
       file "mytrek.com";
       allow-update {key mydhcpserver;};
 };

zone "1.168.192.in-addr.arpa" {
       type master;
       file "192.168.0";
       allow-update {key mydhcpserver;};
};
```

In the DHCP server, you place a corresponding TSIG key declaration and **allow-update** entries in the server's **dhcpd.conf** file, as shown in this example. The **key** declaration has the same syntax as the DNS server. DHCP **zone** statements are then used to specify the IP address of the domain and the TSIG key to use.

```
key mydhcpserver {
    algorithm HMAC-MD5;
    secret "ONQAfbBLnvWU9H8hRqq/WA==";
    };

zone mytrek.com. {                #DNS domain zone to update
       primary 192.168.0.1;       #address of DNS server
       key mydhcpserver;         #TSIG signature key
 };

zone 1.168.192.in-addr.arpa. {    #domain PTR zone to update
       primary 192.168.0.1;       #address of DNS server
       key mydhcpserver;          # TSIG signature key
};
```

The domain names and IP addresses need to match exactly in the configuration files for both the DNS and DHCP servers. Unlike in the **named.conf** file, there are no quotes around the domain name or IP addresses in the **dhcpd.conf** file. In the **dhcpd.conf** file, the domain names and IP addresses used in the **zone** statement also need to end with a period, as they do in the DNS zone files. The **key** statement lists the key to use. Though the DHCP server will try to determine the

DNS servers to update, it is recommended that you explicitly identify them with a primary statement in a **zone** entry.

To generate a fully qualified hostname to use in a DNS update, the DHCP server will normally use its own domain name and the hostname provided by a DHCP client (see the **dhcpd.conf** Man page for exceptions). Should you want to assign a specific hostname to a host, you can use the **ddns-hostname** statement to specify it in the host's hardware section. The domain name is specified in the **domain-name** option:

```
option domain-name "mytrek.com"
```

The DNS update capability can be turned on or off for all domains with the **ddns-update-style** statement. It is on by default. To turn off DNS updates for particular domains, you can use the **ddns-updates** statement. This is also on by default.

DHCP Subnetworks

If you are dividing your network space into several subnetworks, you can use a single DHCP server to manage them. In that case, you will have a **subnet** declaration for each subnetwork. If you are setting up your own small network, you use a network address beginning with 192.168. The range specifies possible IP addresses within that network so, for a network with the address 192.168.0.0, you create a **subnet** declaration with the netmask 255.255.255.0. Within this declaration, you place a **range** declaration along with any other information you want to give to your client hosts. In the following example, a range of IP addresses extending from 192.168.0.1 to 192.168.0.75 can be allocated to the hosts on that network:

```
subnet 192.168.0.0 netmask 255.255.255.0 {
 range 192.168.0.5 192.168.0.75;
}
```

You may want to specify different policies for each subnetwork, such as different lease times. Any entries in a **subnet** declaration will override global settings. So if you already have a global lease time set, a lease setting in a **subnet** declaration will override it for that subnet. The next example sets different lease times for different subnets, as well as different address allocations. The lease times for the first subnet are taken from the global lease time settings, whereas the second subnet defines its own lease times:

```
default-lease-time 21600;
max-lease-time 43200;

subnet 192.168.1.0 netmask 255.255.255.0 {
      range 192.168.0.5 192.168.0.75;
      }
subnet 192.168.1.128 netmask 255.255.255.252 {
      range 192.168.0.129 192.168.0.215;
      default-lease-time 56000;
      max-lease-time 62000;
       }
```

If your subnetworks are part of the same physical network, you need to inform the server of this fact by declaring them as shared networks. You do this by placing subnet declarations within a **shared-network** declaration, specifying the shared network's name. The name can be any descriptive name, though you can use the domain name. Any options specified within the **shared-network** declaration and outside the subnet declarations will be global to those subnets. In the next example, the subnets are part of the same physical network and so are placed within a **shared-network** declaration:

```
shared-network mytrek.com
{
default-lease-time 21600;
max-lease-time 43200;
subnet 192.168.1.0 netmask 255.255.255.0 {
        range 192.168.0.5 192.168.0.75;
        }
subnet 192.168.1.128 netmask 255.255.255.252 {
        range 192.168.0.129 192.168.0.215;
        default-lease-time 56000;
        max-lease-time 62000;
        }
}
```

DHCP Fixed Addresses

Instead of using a pool of possible IP addresses for your hosts, you may want to give each one a specific addresses. Using the DHCP server still gives you control over which address will be assigned to a given host. However, to assign an address to a particular host, you need to know the hardware address for that host's network interface card (NIC). In effect, you have to inform the DHCP server that it has to associate a particular network connection device with a specified IP address. To do that, the DHCP server needs to know which network device you are referring to. You can identify a network device by its hardware address, known as its MAC address. To find out a client's hardware address, you log in to the client and use the **ifconfig** command to find out information about your network devices. To list all network devices, use the **-a** option. If you know your network device name, you can use that. The next example will list all information about the first Ethernet device, **eth0**:

```
ifconfig eth0
```

This will list information on all the client's network connection devices. The entry (usually the first) with the term **HWaddr** will display the MAC address. Once you have the MAC address, you can use it on the DHCP server to assign a specific IP address to that device.

In the **dhcpd.conf** file, you use a **host** declaration to set up a fixed address for a client. Within the **host** declaration, you place a **hardware** option in which you list the type of network connection device and its MAC address. Then you use the **fixed-address** parameter to specify the IP address to be assigned to that device. In the following example, the client's network device with a MAC address of 08:00:2b:4c:29:32 is given the IP address 192.168.0.2:

```
host rabbit {
             option host-name "rabbit.mytrek.com"
             hardware ethernet 08:00:2b:4c:29:32;
             fixed-address 192.168.0.2;
             }
```

You can also have the DHCP server perform a DNS lookup to obtain the host's IP address. This has the advantage of letting you manage IP addresses in only one place, the DNS server. Of course, this requires that the DNS server be operating so that the DHCP server can determine the IP address. For example, a proxy server connection (which can provide direct web access) needs just an IP address, not a DNS hostname, to operate. If the DNS server were down, the preceding example would still assign an IP address to the host, whereas the following example would not:

```
host rabbit {
             option host-name "rabbit.mytrek.com"
             hardware ethernet 08:00:2b:4c:29:32;
             fixed-address rabbit.mytrek.com;
             }
```

You can also use the **host** declaration to define network information for a diskless workstation or terminal. In this case, you add a **filename** parameter specifying the boot file to use for that workstation or terminal. Here the terminal called **myterm** obtains boot information from the server **turtle.mytrek.com**:

```
host myterm {
             option host-name "myterm.mytrek.com"
             filename "/boot/vmlinuz";
             hardware ethernet 08:00:2b:4c:29:32;
             server-name "turtle.mytrek.com";
             }
```

A common candidate for a fixed address is the DNS server for a network. Usually, you want the DNS server located at the same IP address, so that it can be directly accessed. The DHCP server can then provide this IP address to its clients.

fedora

32. Administering TCP/IP Networks

Linux systems are configured to connect into networks that use the TCP/IP protocols. These are the same protocols that the Internet uses, as do many local area networks (LANs). TCP/IP is a robust set of protocols designed to provide communications among systems with different operating systems and hardware. The TCP/IP protocols were developed in the 1970s as a special DARPA project to enhance communications between universities and research centers. These protocols were originally developed on UNIX systems, with much of the research carried out at the University of California, Berkeley. Linux, as a version of UNIX, benefits from much of this original focus on UNIX. Currently, the TCP/IP protocol development is managed by the Internet Engineering Task Force (IETF), which, in turn, is supervised by the Internet Society (ISOC). The ISOC oversees several groups responsible for different areas of Internet development, such as the Internet Assigned Numbers Authority (IANA), which is responsible for Internet addressing (see Table 32-1). Over the years, TCP/IP protocol standards and documentation have been issued in the form of Request for Comments (RFC) documents. Check the most recent ones for current developments at the IETF Web site at **www.ietf.org**.

TCP/IP Protocol Suite

The TCP/IP protocol suite actually consists of different protocols, each designed for a specific task in a TCP/IP network. The three basic protocols are the Transmission Control Protocol (TCP), which handles receiving and sending out communications, the Internet Protocol (IP), which handles the actual transmissions, and the User Datagram Protocol (UDP), which also handles receiving and sending packets. The IP protocol, which is the base protocol that all others use, handles the actual transmissions, handling the packets of data with sender and receiver information in each. The TCP protocol is designed to work with cohesive messages or data. This protocol checks received packets and sorts them into their designated order, forming the original message. For data sent out, the TCP protocol breaks the data into separate packets, designating their order. The UDP protocol, meant to work on a much more raw level, also breaks down data into packets but does not check their order. The TCP/IP protocol is designed to provide stable and reliable connections that ensure that all data is received and reorganized into its original order. UDP, on the other hand, is designed to simply send as much data as possible, with no guarantee that packets will all be received or placed in the proper order. UDP is often used for transmitting very large amounts of data of the type that can survive the loss of a few packets—for example, temporary images, video, and banners displayed on the Internet.

Other protocols provide various network and user services. The Domain Name Service (DNS) provides address resolution. The File Transfer Protocol (FTP) provides file transmission, and the Network File System (NFS) provides access to remote file systems. Table 32-2 lists the different protocols in the TCP/IP protocol suite. These protocols make use of either the TCP or UDP protocol to send and receive packets, which, in turn, uses the IP protocol for actually transmitting the packets.

In a TCP/IP network, messages are broken into small components, called *datagrams,* which are then transmitted through various interlocking routes and delivered to their destination computers. Once received, the datagrams are reassembled into the original message. Datagrams themselves can be broken down into smaller packets. The *packet* is the physical message unit actually transmitted among networks. Sending messages as small components has proved to be far more reliable and faster than sending them as one large, bulky transmission. With small components, if one is lost or damaged, only that component must be resent, whereas if any part of a large transmission is corrupted or lost, the entire message has to be resent.

Group	Title	Description
ISOC	Internet Society	Professional membership organization of Internet experts that oversees boards and task forces dealing with network policy issues www.isoc.org
IESG	The Internet Engineering Steering Group	Responsible for technical management of IETF activities and the Internet standards process www.ietf.org/iesg.html
IANA	Internet Assigned Numbers Authority	Responsible for Internet Protocol (IP) addresses www.iana.org
IAB	Internet Architecture Board	Defines the overall architecture of the Internet, providing guidance and broad direction to the IETF www.iab.org
IETF	Internet Engineering Task Force	Protocol engineering and development arm of the Internet www.ietf.org

Table 32-1: TCP/IP Protocol Development Groups

The configuration of a TCP/IP network on your Linux system is implemented using a set of network configuration files (Table 32-6 provides a complete listing). Many of these can be managed using administrative programs, such as system-config-network on your desktop. You can also use the more specialized programs, such as netstat, ifconfig, Wireshark, and route. Some configuration files are easy to modify yourself using a text editor.

TCP/IP networks are configured and managed with a set of utilities: ifconfig, route, and netstat. The ifconfig utility operates from your root user desktop and enables you to configure your network interfaces fully, adding new ones and modifying others. The ifconfig and route utilities are lower-level programs that require more specific knowledge of your network to use effectively. The netstat utility provides you with information about the status of your network connections. Wireshark is a network protocol analyzer that lets you capture packets as they are transmitted across your network, selecting those you want to check.

Zero Configuration Networking: Avahi and Link Local Addressing

Zero Configuration Networking (Zeroconf) allows the setup of nonroutable private networks without the need of a DHCP server or static IP addresses. A Zeroconf configuration lets users automatically connect to a network and access all network resources, such as printers, without having to perform any configuration. On Linux, Zeroconf networking is implemented by Avahi (http://avahi.org), which includes multicast DNS (mDNS) and DNS service discovery (DNS-SD) support that automatically detects services on a network. IP addresses are determined using either IPv6 or IPv4 Link Local (IPv4LL) addressing. IPv4 Link Local addresses are assigned from the 168.254.0.0 network pool. Derived from Apple's Bonjour Zeroconf implementation, it is a free and open source version currently used by desktop tools such as the GNOME virtual file system. Fedora implements full Zeroconf network support with the Avahi daemon that implements multicast DNS discover, and avahi-autoipd that provides dynamic configuration of local IPv4 addresses. Both are installed as part of the desktop configuration. Avahi support tools like **avahi-**

browse and **avahi-publish** are located in the **avahi-tools** package. Specialized tools like SSH and Shell tools are located in the **avahi-ui-tools** package.

The KDE Zeroconf solution is also provided using Avahi (**kdnssd**) located in the **kdnnsd-avahi** packages. Use the KDE control center Service Discovery panel (Internet & Network section) to specify your domain. Then enter **zeorconf:/** in a KDE file manger window.

IPv4 and IPv6

Traditionally, a TCP/IP address is organized into four segments, consisting of numbers separated by periods. This is called the *IP address.* The IP address actually represents a 32-bit integer whose binary values identify the network and host. This form of IP addressing adheres to Internet Protocol, version 4, also known as IPv4. IPv4, the kind of IP addressing described here, is still in wide use.

Currently, a new version of the IP protocol called Internet Protocol, version 6 (IPv6) is gradually replacing the older IPv4 version. IPv6 expands the number of possible IP addresses by using 128 bits. It is fully compatible with systems still using IPv4. IPv6 addresses are represented differently, using a set of eight 16-bit segments, each separated from the next by a colon. Each segment is represented by a hexadecimal number. A sample address would be

```
FC00:0:0:0:800:BA98:7654:3210
```

Advantages for IPv6 include the following:

- IPv6 features simplified headers that allow for faster processing.
- IPv6 provides support for encryption and authentication along with virtual private networks (VPN) using the integrated IPsec protocol.
- One of its most significant advantages lies in its extending the address space to cover 2 to the power of 128 possible hosts (billions of billions). This extends far beyond the 4.2 billion supported by IPv4.
- IPv6 supports stateless autoconfiguration of addresses for hosts, bypassing the need for DHCP to configure such addresses. Addresses can be generated directly using the MAC (Media Access Control) hardware address of an interface.
- IPv6 provides support for Quality of Service (QoS) operations, providing sufficient response times for services like multimedia and telecom tasks.
- Multicast capabilities are built into the protocol, providing direct support for multimedia tasks. Multicast addressing also provides that same function as IPv4 broadcast addressing.
- More robust transmissions can be ensured with anycast addressing, where packets can be directed to an anycast group of systems, only one of which needs to received them. Multiple DNS servers supporting a given network could be designated as an anycast group, of which only one DNS server needs to receive the transmission, providing greater likelihood that the transmissions will go through.
- IPv6 provides better access for mobile nodes, like PDAs, notebooks, and cell phones.

Transport	**Description**
TCP	Transmission Control Protocol; places systems in direct communication
UDP	User Datagram Protocol
IP	Internet Protocol; transmits data
ICMP	Internet Control Message Protocol; status messages for IP
Routing	**Description**
RIP	Routing Information Protocol; determines routing
OSPF	Open Shortest Path First; determines routing
Network Addresses	**Description**
ARP	Address Resolution Protocol; determines unique IP address of systems
DNS	Domain Name Service; translates hostnames into IP addresses
RARP	Reverse Address Resolution Protocol; determines addresses of systems
User Service	**Description**
FTP	File Transfer Protocol; transmits files from one system to another using TCP
TFTP	Trivial File Transfer Protocol; transfers files using UDP
Telnet	Remote login to another system on the network
SMTP	Simple Mail Transfer Protocol; transfers e-mail between systems
RPC	Remote Procedure Call; allows programs on remote systems to communicate
Gateway	**Description**
EGP	Exterior Gateway Protocol; provides routing for external networks
GGP	Gateway-to-Gateway Protocol; provides routing between Internet gateways
IGP	Interior Gateway Protocol; provides routing for internal networks
Network Service	**Description**
NFS	Network File System; allows mounting of file systems on remote machines
NIS	Network Information Service; maintains user accounts across a network
BOOTP	Boot Protocol; starts system using boot information on server for network
SNMP	Simple Network Management Protocol; provides status messages on TCP/IP configuration
DHCP	Dynamic Host Configuration Protocol; automatically provides network configuration information to host systems

Table 32-2: TCP/IP Protocol Suite

TCP/IP Network Addresses

As noted previously, the traditional IPv4 TCP/IP address is organized into four segments, consisting of numbers separated by periods. This kind of address is still in widespread use and is what people commonly refer to as an *IP address.* Part of an IP address is used for the network address, and the other part is used to identify a particular interface on a host in that network. You should realize that IP addresses are assigned to interfaces—such as Ethernet cards or modems—and not to the host computer. Usually a computer has only one interface and is accessed using only that interface's IP address. In that regard, an IP address can be thought of as identifying a particular host system on a network, and so the IP address is usually referred to as the *host address.*

In fact, though, a host system could have several interfaces, each with its own IP address. This is the case for computers that operate as gateways and firewalls from the local network to the Internet. One interface usually connects to the LAN and another to the Internet, as by two Ethernet cards. Each interface (such as an Ethernet card) has its own IP address. For example, when you use the system-config-network tool to specify an IP address for an Ethernet card on your system, the Devices tab lists an entry for each Ethernet card installed on your computer, beginning with **eth0** for the first. Opening up a Device window, you can select the TCP protocol in the Protocols tab to open a TCP/IP setting window where you can enter the card's IP address. Other Ethernet cards have their own IP addresses. If you use a modem to connect to an ISP, you would set up a Point-to-Point (PPP) interface that would also have its own IP address (usually dynamically assigned by the ISP). This distinction is important if you plan to use Linux to set up a local or home network, using Linux as your gateway machine to the Internet.

IPv4 Network Addresses

The IP address is divided into two parts: one part identifies the network, and the other part identifies a particular host. The network address identifies the network of which a particular interface on a host is a part. Two methods exist for implementing the network and host parts of an IP address: the original class-based IP addressing and the current Classless Interdomain Routing (CIDR) addressing. Class-based IP addressing designates officially predetermined parts of the address for the network and host addresses, whereas CIDR addressing allows the parts to be determined dynamically using a netmask.

Class-Based IP Addressing

Originally, IP addresses were organized according to classes. On the Internet, networks are organized into three classes depending on their size—classes A, B, and C. A class A network uses only the first segment for the network address and the remaining three for the host, allowing a great many computers to be connected to the same network. Most IP addresses reference smaller, class C, networks. For a class C network, the first three segments are used to identify the network, and only the last segment identifies the host. Altogether, this forms a unique address with which to identify any network interface on computers in a TCP/IP network. For example, in the IP address 192.168.1.72, the network part is 192.168.1 and the interface/host part is 72. The interface/host is a part of a network whose own address is 192.168.1.0.

In a class C network, the first three numbers identify the network part of the IP address. This part is divided into three network numbers, each identifying a subnet. Networks on the Internet are organized into subnets, beginning with the largest and narrowing to small subnetworks. The last number is used to identify a particular computer, referred to as a *host.* You can think of the

Internet as a series of networks with subnetworks; these subnetworks have their own subnetworks. The rightmost number identifies the host computer, and the number preceding it identifies the subnetwork of which the computer is a part. The number to the left of that identifies the network the subnetwork is part of, and so on. The Internet address 192.168.187.4 references the fourth computer connected to the network identified by the number 187. Network 187 is a subnet to a larger network identified as 168. This larger network is itself a subnet of the network identified as 192. Here's how it breaks down:

192.168.187.4	IPv4 address
192.168.187	Network identification
4	Host identification

Netmask

Systems derive the network address from the host address using the netmask. You can think of an IP address as a series of 32 binary bits, some of which are used for the network and the remainder for the host. The *netmask* has the network set of bits set to 1s, with the host bits set to 0s (see Figure 32-1). In a standard class-based IP address, all the numbers in the network part of your host address are set to 255, and the host part is set to 0. This has the effect of setting all the binary bits making up the network address to 1s. This, then, is your netmask. So, the netmask for the host address 192.168.1.72 is 255.255.255.0. The network part, 192.168.1, has been set to 255.255.255, and the host part, 72, has been set to 0. Systems can then use your netmask to derive your network address from your host address. They can determine what part of your host address makes up your network address and what those numbers are.

For those familiar with computer programming, a bitwise AND operation on the netmask and the host address results in zeroing the host part, leaving you with the network part of the host address. You can think of the address as being implemented as a four-byte integer, with each byte corresponding to a segment of the address. In a class C address, the three network segments correspond to the first three bytes and the host segment corresponds to the fourth byte. A netmask is designed to mask out the host part of the address, leaving the network segments alone. In the netmask for a standard class C network, the first three bytes are all 1s and the last byte consists of 0s. The 0s in the last byte mask out the host part of the address, and the 1s in the first three bytes leave the network part of the address alone. Figure 32-1 shows the bitwise operation of the netmask on the address 192.168.1.4. This is a class C address to the mask, which consists of twenty-four 1s making up the first three bytes and eight 0s making up the last byte. When it is applied to the address 192.168.1.4, the network address remains (192.168.1) and the host address is masked out (4), giving you 192.168.1.0 as the network address.

The netmask as used in Classless Interdomain Routing (CIDR) is much more flexible. Instead of having the size of the network address and its mask determined by the network class, it is determined by a number attached to the end of the IP address. This number simply specifies the size of the network address, how many bits in the address it takes up. For example, in an IP address whose network part takes up the first three bytes (segments), the number of bits used for that network part is 24—eight bits to a byte (segment). Instead of using a netmask to determine the network address, the number for the network size is attached to the end of the address with a slash, as shown here:

```
192.168.1.72/24
```

CIDR gives you the advantage of specifying networks that are any size bits, instead of only three possible segments. You could have a network whose addresses take up 14 bits, 22 bits, or even 25 bits. The host address can use whatever bits are left over. An IP address with 21 bits for the network can cover host addresses using the remaining 11 bits, 0 to 2,047.

Classless Interdomain Routing (CIDR)

Currently, the class-based organization of IP addresses is being replaced by the CIDR format. CIDR was designed for midsized networks, those between a class C and classes with numbers of hosts greater than 256 and smaller than 65,534. A class C network–based IP address uses only one segment, an 8-bit integer, with a maximum value of 256. A class B network–based IP address uses two segments, which make up a 16-bit integer whose maximum value is 65,534. You can think of an address as a 32-bit integer taking up four bytes, where each byte is 8 bits. Each segment conforms to one of the four bytes. A class C network uses three segments, or 24 bits, to make up its network address. A class B network, in turn, uses two segments, or 16 bits, for its address. With this scheme, allowable host and network addresses are changed an entire byte at a time, segment to segment. With CIDR addressing, you can define host and network addresses by bits, instead of whole segments. For example, you can use CIDR addressing to expand the host segment from 8 bits to 9, rather than having to jump it to a class B 16 bits (two segments).

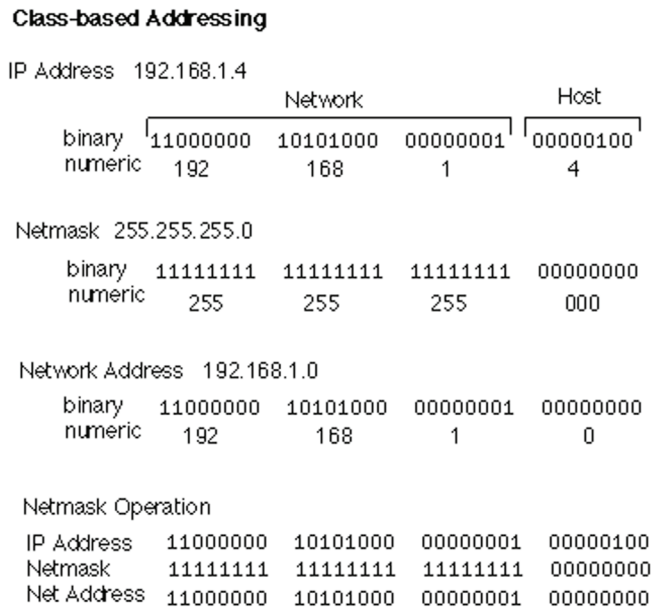


Figure 32-1: Class-based netmask operations

CIDR addressing notation achieves this by incorporating netmask information in the IP address (the netmask is applied to an IP address to determine the network part of the address). In

the CIDR notation, the number of bits making up the network address is placed after the IP address, following a slash. For example, the CIDR form of the class C 192.168.187.4 IP address is

```
192.168.187.4/24
```

Figure 32-2 shows an example of a CIDR address and its network mask. The IP address is 192.168.1.6 with a network mask of 22 bits, 192.168.1.6/22. The network address takes up the first 22 bits of the IP address, and the remaining 10 bits are used for the host address. The host address is taking up the equivalent of a class-based IP address's fourth segment (8 bits) and 2 bits from the third segment.

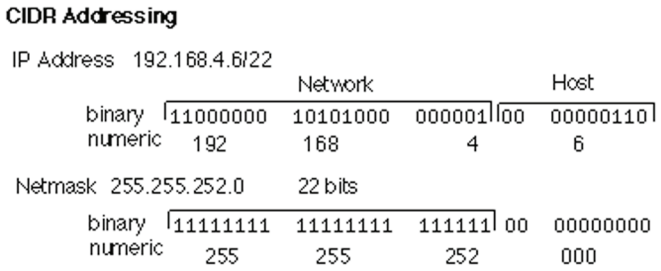


Figure 32-2: CIDR addressing

Table 32-3 lists the different IPv4 CIDR network masks available along with the maximum number of hosts. Both the short forms and the full forms of the netmasks are listed.

IPv4 CIDR Addressing

The network address for any standard class C IPv4 IP address takes up the first three segments, 24 bits. If you want to create a network with a maximum of 512 hosts, you can give them IP addresses where the network address is 23 bits and the host address takes up 9 bits (0–511). The IP address notation remains the same, however, using the four 8-bit segments. This means a given segment's number could be used for both a network address and a host address. Segments are no longer wholly part of either the host address or the network address. Assigning a 23-bit network address and a 9-bit host address means that the number in the third segment is part of the network address and the host address, the first 7 bits for the network and the last bit for the host. In this following example, the third number, 145, is used as the end of the network address and as the beginning of the host address:

```
192.168.145.67/23
```

This situation complicates CIDR addressing, and in some cases the only way to represent the address is to specify two or more network addresses. Check RFC 1520 at **www.ietf.org** for more details.

CIDR also allows a network administrator to take what is officially the host part of an IP address and break it up into subnetworks with fewer hosts. This is referred to as *subnetting*. A given network will have its official IP network address recognized on the Internet or by a larger network. The network administrator for that network could, in turn, create several smaller networks within it using CIDR network masking. A classic example is to take a standard class C network with 254 hosts and break it up into two smaller networks, each with 64 hosts. You do this by using a CIDR netmask to take a bit from the host part of the IP address and use it for the subnetworks.

Numbers within the range of the original 254 addresses whose first bit would be set to 1 would represent one subnet, and the others, whose first bit would be set to 0, would constitute the remaining network. In the network whose network address is 192.168.187.0, where the last segment is used for the hostnames, that last host segment could be further split into two subnets, each with its own hosts. For two subnets, you would use the first bit in the last 8-bit segment for the network. The remaining 7 bits could then be used for host addresses, giving you a range of 127 hosts per network. The subnet whose bit is set to 0 would have a range of 1 to 127, with a CIDR netmask of 25. The 8-bit segment for the first host would be 00000001. So the host with the address of 1 in that network would have this IP address:

```
192.168.187.1/25
```

Short Form	Full Form	Maximum Number of Hosts
/8	/255.0.0.0	16,777,215 (A class)
/16	/255.255.0.0	65,535 (B class)
/17	/255.255.128.0	32,767
/18	/255.255.192.0	16,383
/19	/255.255.224.0	8,191
/20	/255.255.240.0	4,095
/21	/255.255.248.0	2,047
/22	/255.255.252.0	1,023
/23	/255.255.254.0	511
/24	/255.255.255.0	255 (C class)
/25	/255.255.255.128	127
/26	/255.255.255.192	63
/27	/255.255.255.224	31
/28	/255.255.255.240	15
/29	/255.255.255.248	7
/30	/255.255.255.252	3

Table 32-3: CIDR IPv4 Network Masks

For the subnet where the first bit is 1, the first host would have an address of 129, with the CIDR netmask of 25, as shown here. The 8-bit sequence for the first host would be 10000001.

```
192.168.187.129/25
```

Note: A simple way to calculate the number of hosts a network can address is to take the number of bits in its host segment as a power of 2, and then subtract 2—that is, 2 to the number of host bits, minus 2. For example, an 8-bit host segment would be 2 to the power of 8, which equals 256. Subtract 2 (1 for the broadcast address, 255, and 1 for the zero value, 000) to leave you with 254 possible hosts.

Each subnet would have a set of 126 addresses, the first from 1 to 126, and the second from 129 to 254; 127 is the broadcast address for the first subnet, and 128 is the network address for the second subnet. The possible subnets and their masks that you could use are shown here:

Subnetwork	CIDR Address	Binary Mask
First subnet network address	.0/25	00000000
Second subnet network address	.128/25	10000000
First subnet broadcast address	.127/25	01111111
Second subnet broadcast address	.255/25	11111111
First address in first subnet	.1/25	00000001
First address in second subnet	.129/25	10000001
Last address in first subnet	.126/25	01111110
Last address in second subnet	.254/25	11111110

IPv6 CIDR Addressing

IPv6 CIDR addressing works much the same as with the IPv4 method. The number of bits used for the network information is indicated by number following the address. A host (interface) address could take up much more than the 64 bits that it usually does in an IPv6 address, making the network prefix (address) section smaller than 64 bits. How many bits that the network prefix uses is indicated by the following number. In the next example the network prefix (address) uses only the first 48 bits of the IPv6 address, and the host address uses the remaining 80 bits:

```
FC00:0000:0000:0000:FEDC:BA98:7654:3210/48
```

You can also use a two-colon notation (::) for the compressed version:

```
FC00::FEDC:BA98:7654:3210/48
```

Though you can use CIDR to subnet addresses, IPv6 also supports a subnet field that can be used for subnets.

Obtaining an IP Address

IP addresses are officially allocated by IANA, which manages all aspects of Internet addressing (**www.iana.org**). IANA oversees Internet Registries, which, in turn, maintain Internet addresses on regional and local levels. The Internet Registry for the Americas is the American Registry for Internet Numbers (ARIN), whose Web site is at **www.arin.net**. These addresses are provided to users by Internet service providers (ISPs). You can obtain your own Internet address from an ISP, or if you are on a network already connected to the Internet, your network administrator can assign you one. If you are using an ISP, the ISP may temporarily assign one from a pool it has on hand with each use.

IPv4 Reserved Addresses

Certain numbers are reserved. The numbers 127, 0, or 255 cannot be part of an official IP address. The number 127 is used to designate the network address for the loopback interface on your system. The loopback interface enables users on your system to communicate with each other within the system without having to route through a network connection. Its network address would be 127.0.0.0, and its IP address is 127.0.0.1. For class-based IP addressing, the number 255 is a special broadcast identifier you can use to broadcast messages to all sites on a network. Using 255 for any part of the IP address references all nodes connected at that level. For example, 192.168.255.255 broadcasts a message to all computers on network 192.168, all its subnetworks, and their hosts. The address 192.168.187.255 broadcasts to every computer on the local network. If you use 0 for the network part of the address, the host number references a computer within your local network. For example, 0.0.0.6 references the sixth computer in your local network. If you want to broadcast to all computers on your local network, you can use the number 0.0.0.255. For CIDR IP addressing, the broadcast address may appear much like a normal IP address. As indicated in the preceding section, CIDR addressing allows the use of any number of bits to make up the IP address for either the network or the host part. For a broadcast address, the host part must have all its bits set to 1 (see Figure 32-3).

A special set of numbers is reserved for use on non-Internet LANs (RFC 1918). These are numbers that begin with the special network number 192.168 (for class C networks), as used in these examples. If you are setting up a LAN, such as a small business or a home network, you are free to use these numbers for your local machines. You can set up an intranet using network cards, such as Ethernet cards and Ethernet hubs, and then configure your machines with IP addresses starting from 192.168.1.1. The host segment can go up to 256. If you have three machines on your home network, you could give them the addresses 192.168.1.1, 192.168.1.2, and 192.168.1.3. You can implement Internet services, such as FTP, Web, and mail services, on your local machines and use any of the Internet tools to make use of those services. They all use the same TCP/IP protocols used on the Internet. For example, with FTP tools, you can transfer files among the machines on your network. With mail tools, you can send messages from one machine to another, and with a Web browser, you can access local Web sites that may be installed on a machine running its own Web servers. If you want to have one of your machines connected to the Internet or some other network, you can set it up to be a gateway machine. By convention, the gateway machine is usually given the address 192.168.1.1. With a method called *IP masquerading,* you can have any of the non-Internet machines use a gateway to connect to the Internet.

IPv4 Private Network Addresses	Network Classes
10.0.0.0	Class A network
172.16.0.0–172.31.255.255	Class B network
192.168.0.0	Class C network
127.0.0.0	Loopback network (for system self-communication)

Table 32-4: Non-Internet IPv4 Local Network IP Addresses

Numbers are also reserved for class A and class B non-Internet local networks. Table 32-4 lists these addresses. The possible addresses available span from 0 to 255 in the host segment of the address. For example, class B network addresses range from 172.16.0.0 to 172.31.255.255, giving you a total of 32,356 possible hosts. The class C network ranges from 192.168.0.0 to

192.168.255.255, giving you 256 possible subnetworks, each with 256 possible hosts. The network address 127.0.0.0 is reserved for a system's loopback interface, which allows it to communicate with itself, enabling users on the same system to send messages to each other.

Broadcast Addresses

The broadcast address allows a system to send the same message to all systems on your network at once. With IPv4 class-based IP addressing, you can easily determine the broadcast address using your host address: the broadcast address has the host part of your address set to 255. The network part remains untouched. So the broadcast address for the host address 192.168.1.72 is 192.168.1.255 (you combine the network part of the address with 255 in the host part). For CIDR IP addressing, you need to know the number of bits in the netmask. The remaining bits are set to 1 (see Figure 32-3). For example, an IP address of 192.168.4.6/22 has a broadcast address of 192.168.7.255/22. In this case, the first 22 bits are the network address and the last 10 bits are the host part set to the broadcast value (all 1s).

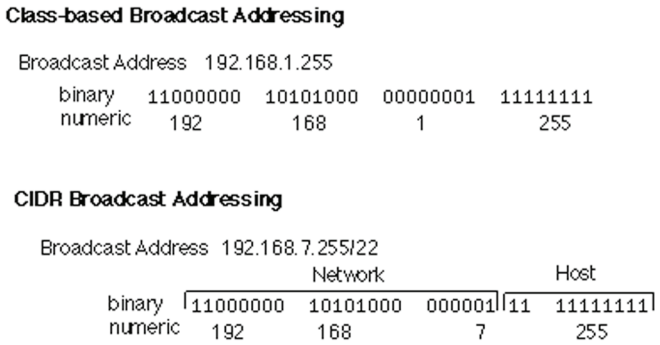


Figure 32-3: Class-based and CIDR broadcast addressing

In fact, you can think of a class C broadcast address as merely a CIDR address using 24 bits (the first three segments) for the network address, and the last 8 bits (the fourth segment) as the broadcast address. The value 255 expressed in binary terms is simply 8 bits that are all 1s. 255 is the same as 11111111.

IP Address	Broadcast Address	IP Broadcast Number	Binary Equivalent
192.168.1.72	192.168.1.255	255	11111111
192.168.4.6/22	192.168.7.255/22	7.255 (last 2 bits in 7)	1111111111

Gateway Addresses

Some networks have a computer designated as the gateway to other networks. Every connection to and from a network to other networks passes through this gateway computer. Most local networks use gateways to establish a connection to the Internet. If you are on this type of network, you must provide the gateway address. If your network does not have a connection to the Internet, or a larger network, you may not need a gateway address. The gateway address is the address of the host system providing the gateway service to the network. On many networks, this host is given a host ID of 1: the gateway address for a network with the address 192.168.1 would be

192.168.1.1, but this is only a convention. To be sure of your gateway address, ask your network administrator.

Name Server Addresses

Many networks, including the Internet, have computers that provide a Domain Name Service (DNS) that translates the domain names of networks and hosts into IP addresses. These are known as the network's *domain name servers.* The DNS makes your computer identifiable on a network, using only your domain name, rather than your IP address. You can also use the domain names of other systems to reference them, so you needn't know their IP addresses. You must know the IP addresses of any domain name servers for your network, however. You can obtain the addresses from your system administrator (often more than one exists). Even if you are using an ISP, you must know the address of the domain name servers your ISP operates for the Internet.

IPv6 Addressing

IPv6 addresses introduces major changes into the format and method of addressing systems under the Internet Protocol (see RFC 3513 at **www.ietf.org/rfc** or **www.faqs.org** for more details). There are several different kinds of addressing with different fields for the network segment. The host segment has been expanded to a 64-bit address, allowing direct addressing for a far larger number of systems. Each address begins with a type field specifying the kind of address, which will then determine how its network segment is organized. These changes are designed not only to expand the address space but to also provide greater control over transmissions at the address level.

Note: Red Hat Enterprise Linux and Fedora are distributed with IPv6 support already enabled in the kernel. Kernel support for IPv6 is provided by the IPv6 kernel module. Kernel configuration support can be found under Device Drivers | Networking Support | Networking Options | The IPv6 Protocol.

IPv6 Address Format

An IPv6 address consists of 128 bits, up from the 32 bits used in IPv4 addresses. The first 64 bits are used for network addressing, of which the first few bits are reserved for indicating the address type. The last 64 bits are used for the interface address, known as the interface identifier field. The amount of bits used for subnetting can be adjusted with a CIDR mask, much like that in IPv4 CIDR addressing (see the preceding section).

An IPv6 address is written as eight segments representing 16 bits each (128 bits total). To more easily represent 16-bit binary numbers, hexadecimal numbers are used. Hexadecimal numbers use 16 unique numbers, instead of the 8 used in octal numbering. These are 0–9, continuing with the characters A–F.

In the following example the first four segments represent the network part of the IPv6 address, and the following four segments represent the interface (host) address:

```
FC00:0000:0000:0000:0008:0800:200C:417A
```

You can cut any preceding zeros, but not trailing zeros, in any given segment. Segments with all zeros can be reduced to a single zero.

```
FC00:0:0:0:8:800:200C:417A
```

The loopback address used for localhost addressing can be written with seven preceding zeros and a 1.

```
0:0:0:0:0:0:0:1
```

Many addresses will have sequences of zeros. IPv6 supports a shorthand symbol for representing a sequence of several zeros in adjacent fields. This consists of a double colon (::). There can be only one use of the :: symbol per address.

```
FC00::8:800:200C:417A
```

The loopback address 0000000000000001 can be reduced to just the following:

```
::1
```

To ease the transition from IPv4 addressing to IPv6, a form of addressing incorporating IPv4 addresses is also supported. In this case, the IPv4 address (32 bits) can be used to represent the last two segments of an IPv6 address and can be written using IPv4 notation.

```
FC00::192.168.0.3
```

IPv6 Interface Identifiers

The identifier part of the IPv6 address takes up the second 64 bits, consisting of four segments containing four hexadecimal numbers. The interface ID is a 64-bit (four-segment) Extended Unique Identifier (EUI-64) generated from a network device's Media Access Control (MAC) address.

IPv6 Address types

There are three basic kinds of IPv6 addresses, unicast, multicast, and anycast.

- A *unicast* address is used for a packet that is sent to a single destination.
- An *anycast* address is used for a packet that can be sent to more than one destination.
- A *multicast* address is used to broadcast a packet to a range of destinations.

In IPv6, addressing is controlled by the format prefix that operates as a kind of address type. The format prefix is the first field of the IP address. The three major kinds of unicast network addresses are global, link-local, and unique-local. Global, unique-local, and link-local are indicated by their own format prefix (see Table 32-5).

- Global addresses begin with the address type 3, unique-local with FE00, and link-local with FE8. Global addresses can be sent across the Internet.
- Link-local addresses are used for physically connected systems on a local network. It is often used for DHCP addresses.
- Unique-local can be used for any hosts on a local network. Unique-local addresses operate like IPv4 private addresses; they are used only for local access and cannot be used to transmit over the Internet.

In addition, IPv6 has two special reserved addresses. The address 0000000000000001 is reserved for the loopback address used for a system's localhost address, and the address 0000000000000000 is the unspecified address.

IPv6 Addresses Format Prefixes and Reserved Addresses	Description
3	Unicast global addresses
FE8	Unicast link-local addresses, used for physically connected hosts on a network, used for DHCP equivalents.
FC00	Unicast unique-local addresses, comparable to IPv4 private addresses.
0000000000000001	Unicast loopback address (for system self-communication, localhost)
0000000000000000	Unspecified address
FF	Multicast addresses

Table 32-5: IPv6 Format Prefixes and Reserved Addresses

IPv6 Unicast Global Addresses

IPv6 global addresses currently use four fields: the format prefix, a global routing prefix, the subnet identifier, and the interface identifier. The format prefix for a unicast global address is 3 (3 bits). The global routing prefix references the network address (45 bits), and the subnet ID references a subnet within the site (16 bits).

IPv6 Unicast Local Use Addresses: Link-Local and Unique-Local Addresses

For local use, IPv6 provides both link-local and unique-local addresses. Link-local addressing is used for interfaces (hosts) that are physically connected to a network. This is usually a small local network. A link-local address uses only three fields, the format prefix **FE8** (10 bits), an empty field (54 bits), and the interface identifier (host address) (64 bits). In effect, the network section is empty.

IPv6 unique-local addresses have three fields: the format prefix (10 bits), the subnet identifier (54 bits), and the interface identifier (64 bits). Except for any local subnetting, there is no network address. The unique local address has a format prefix of **FC00**. The unique-local addresses (also known as unique local addresses) fulfill the same function as private addresses in IPv4 (192.168.0).

IPv6 Multicast Addresses

Multicast addresses have a format prefix of FF (8 bits) with flag and scope fields to indicate whether the multicast group is permanent or temporary and whether it is local or global in scope. A group identifier (112 bits) references the multicast group. For the scope, 2 is link-local, 5 is unique-local, and E is global. In addition to their interface identifiers, hosts will also have a group ID that can be used as a broadcast address. You use this address to broadcast to the hosts. The following example will broadcast only to those hosts on the local network (5) with the group ID 101:

```
FF05:0:0:0:0:0:0:101
```

To broadcast to all the hosts in a link-local scope, you would use the broadcast address:

```
FF02:0:0:0:0:0:0:1
```

For a unique-local scope, a local network, you would use

```
FF05:0:0:0:0:0:0:2
```

IPv6 and IPv4 Coexistence Methods

In the transition from IPv4 to IPv6, many networks will find the need to support both. Some will be connected to networks that use the contrary protocol, and others will connect through other network connections that use that protocol. There are several official IETF methods for providing IPv6 and IPv4 cooperation, which fall into three main categories:

- **Dual-stack** Allows IPv4 and IPv6 to coexist on the same networks.
- **Translation** Enables IPv6 devices to communicate with IPv4 devices.
- **Tunneling**: Allows transmission from one IPv6 network to another through IPv4 networks as well as allowing IPv6 hosts to operate on or through IPv4 networks.

In the dual-stack methods both IPv6 and IPv4 addresses are supported on the network. Applications and DNS servers can use either to transmit data.

Translation uses NAT tables to translate IPv6 addresses to corresponding IPv4 address and vice versa as needed. IPv4 applications can then freely interact with IPv6 applications. IPv6-to-IPv6 transmissions are passed directly through, enabling full IPv6 functionality.

Tunneling is used when one IPv6 network needs to transmit to another through an IPv4 network that cannot handle IPv6 addresses. With tunneling, the IPv6 packet is encapsulated within an IPv4 packet, where the IPv4 network then uses the outer IPv4 addressing to pass on the packet. Several methods are used for tunneling, as shown here, as well as direct manual manipulation:

- **6-over-4** Used within a network to use IPv4 multicasting to implement a virtual LAN to support IPv6 hosts, without an IPv6 router (RFC 2529)
- **6-to-4** Used to allow IPv6 networks to connect to and through a larger IPv4 network (the Internet), using the IPv4 network address as an IPv6 network prefix (RFC 3056)
- **Tunnel brokers** Web-based services that create tunnels (RFC 3053)

TCP/IP Configuration Files

A set of configuration files in the **/etc** directory, shown in Table 32-6, are used to set up and manage your TCP/IP network. These configuration files specify such network information as host and domain names, IP addresses, and interface options. The IP addresses and domain names of other Internet hosts you want to access are entered in these files. If you configured your network during installation, you can already find that information in these files.

Identifying Hostnames: /etc/hosts

Without the unique IP address the TCP/IP network uses to identify computers, a particular computer cannot be located. Because IP addresses are difficult to use or remember, domain names are used instead. For each IP address, a domain name exists. When you use a domain name to

reference a computer on the network, your system translates it into its associated IP address. This address can then be used by your network to locate that computer.

Address	Description
Host address	IP address of your system; it has a network part to identify the network you are on and a host part to identify your own system
Network address	IP address of your network
Broadcast address	IP address for sending messages to all hosts on your network at once
Gateway address	IP address of your gateway system, if you have one (usually the network part of your host IP address with the host part set to 1)
Domain name server addresses	IP addresses of domain name servers your network uses
Netmask	Used to determine the network and host parts of your IP address
File	**Description**
/etc/hosts	Associates hostnames with IP addresses, lists domain names for remote hosts with their IP addresses
/etc/sysconfig/network-scripts	Network connection configurations
/etc/host.conf	Lists resolver options
/etc/nsswitch.conf	Name Switch Service configuration
/etc/resolv.conf	Lists domain name server names, IP addresses (nameserver), and domain names where remote hosts may be located (search)
/etc/protocols	Lists protocols available on your system
/etc/services	Lists available network services, such as FTP and Telnet, and the ports they use
/etc/sysconfig/networking	Holds network configuration files managed by system-config-network
/etc/sysconfig/network	Network configuration information

Table 32-6: TCP/IP Configuration Addresses and Files

Originally, every computer on the network was responsible for maintaining a list of the hostnames and their IP addresses. This list is still kept in the **/etc/hosts** file. When you use a domain name, your system looks up its IP address in the **hosts** file. The system administrator is responsible for maintaining this list. Because of the explosive growth of the Internet and the development of larger networks, the responsibility for associating domain names and IP addresses has been taken over by domain name servers. The **hosts** file is still used to hold the domain names and IP addresses of frequently accessed hosts, however. Your system normally checks your **hosts** file for the IP address of a domain name before taking the added step of accessing a name server.

The format of a domain name entry in the **hosts** file is the IP address followed by the domain name, separated by a space. You can then add aliases for the hostname. After the entry, on

the same line, you can enter a comment. A comment is always preceded by a **#** symbol. You can already find an entry in your **hosts** file for localhost.localdomain and localhost with the IP address 127.0.0.1; localhost is a special identification used by your computer to enable users on your system to communicate locally with each other. The IP address 127.0.0.1 is a special reserved address used by every computer for this purpose. It identifies what is technically referred to as a *loopback device.* The corresponding IPV6 localhost address is **::1** which has the host name **localhost6**. You should never remove the **localhost** and **localhost6** entries. A sample **/etc/hosts** file is shown here:

/etc/hosts

```
127.0.0.1           localhost.localdomain localhost turtle.mytrek.com
::1                 localhost6.localdomain6  localhost6
192.168.0.1         turtle.mytrek.com
192.168.0.2         rabbit.mytrek.com
192.168.34.56       pango1.mytrain.com
```

/etc/resolv.conf

The **/etc/resolv.conf** file holds the IP addresses for your DNS servers along with domains to search. A DNS entry will begin with the term nameserver followed by the name server's IP address. A search entry will list network domain addresses. Check this file to see if your network DNS servers have been correctly listed. I you have a router for a local network, DHCP will automatically place an entry for it in the in this file. The router in turn will reference your ISP's nameserver.

/etc/resolv.conf

```
search  mytrek.com   mytrain.com
nameserver  192.168.0.1
nameserver  192.168.0.3
```

/etc/sysconfig/network-scripts

The **/etc/sysconfig/network-scripts** directory holds configuration information for different network connection devices such the IP address and network address used. For a detailed discussion, see the section "Network Interfaces and Routes: ifconfig and route" later in this chapter.

/etc/sysconfig/networking

The **/etc/sysconfig/networking** directory holds configuration information set up with system-config-network (Network on the System Settings menu and window). These file should not be edited manually. The profiles directory holds configurations for the different profiles you set up. Different profiles directories will include the hosts file listing host domain names and IP addresses, the network file holding your system's host name, and the **resolv.conf** file, which contains your domain name servers. The device configuration file for the connection you use for that profile will also be listed, such as **ifcfg-eth0** for the first Ethernet device. These are all configurations that may change depending on the profile you use. For example, at the office you may use an Ethernet connection on a company network with its own DNS servers, whereas at home you may use a modem connection to an ISP with its own Internet DNS servers. Your hostname and domain name may vary depending on the networks your different profiles connect to.

/etc/services

The **/etc/services** file lists network services available on your system, such as FTP and Telnet, and associates each with a particular port. Here, you can find out what port your Web server is checking or what port is used for your FTP server. You can give a service an alias, which you specify after the port number. You can then reference the service using the alias.

/etc/protocols

The **/etc/protocols** file lists the TCP/IP protocols currently supported by your system. Each entry shows the protocol number, its keyword identifier, and a brief description. See **www.iana.org/assignments/protocol-numbers** for a complete listing.

/etc/sysconfig/network

The **/etc/sysconfig/network** file contains system definitions for your network configuration. These include definitions for your domain name, gateway, and hostname, as shown here:

```
NETWORKING=yes
HOSTNAME=turtle.mytrek.com
GATEWAY=192.168.0.1
```

Domain Name System (DNS)

Each computer connected to a TCP/IP network, such as the Internet, is identified by its own IP address. IP addresses are difficult to remember, so a domain name version of each IP address is also used to identify a host. A domain name consists of two parts, the hostname and the domain. The hostname is the computer's specific name, and the domain identifies the network of which the computer is a part. The domains used for the United States usually have extensions that identify the type of host. For example, **.edu** is used for educational institutions and **.com** is used for businesses. International domains usually have extensions that indicate the country they are located in, such as **.fr** for France or **.au** for Australia. The combination of a hostname, domain, and extension forms a unique name by which a computer can be referenced. The domain can, in turn, be split into further subdomains.

As you know, a computer on a network can still be identified only by its IP address, even if it has a hostname. You can use a hostname to reference a computer on a network, but this involves using the hostname to look up the corresponding IP address in a database. The network then uses the IP address, not the hostname, to access the computer. Before the advent of large TCP/IP networks, such as the Internet, it was feasible for each computer on a network to maintain a file with a list of all the hostnames and IP addresses of the computers connected on its network. Whenever a hostname was used, it was looked up in this file and the corresponding IP address was located. You can still do this on your own system for remote systems you access frequently.

As networks became larger, it became impractical—and, in the case of the Internet, impossible—for each computer to maintain its own list of all the domain names and IP addresses. To provide the service of translating domain addresses to IP addresses, databases of domain names were developed and placed on their own servers. To find the IP address of a domain name, you send a query to a name server, which then looks up the IP address for you and sends it back. In a large network, several name servers can cover different parts of the network. If a name server

cannot find a particular IP address, it sends the query on to another name server that is more likely to have it.

If you are administering a network and you need to set up a name server for it, you can configure a Linux system to operate as a name server. To do so, you must start up a name server daemon and then wait for domain name queries. A name server makes use of several configuration files that enable it to answer requests. The name server software used on Linux systems is the Berkeley Internet Name Domain (BIND) server distributed by the Internet Software Consortium (**www.isc.org**). Chapter 30 describes the process of setting up a domain name server in detail.

Name servers are queried by resolvers. These are programs specially designed to obtain addresses from name servers. To use domain names on your system, you must configure your own resolver. Your local resolver is configured with your **/etc/host.conf** and **/etc/resolv.conf** files. You can use **/etc/nsswitch** in place of **/etc/host.conf**.

host.conf

Your **host.conf** file lists resolver options (shown in Table 32-7). Each option can have several fields, separated by spaces or tabs. You can use a **#** at the beginning of a line to enter a comment. The options tell the resolver what services to use. The order of the list is important. The resolver begins with the first option listed and moves on to the next ones in turn. You can find the **host.conf** file in your **/etc** directory, along with other configuration files.

Option	Description
order	Specifies sequence of name resolution methods: **hosts** Checks for name in the local **/etc/host** file **bind** Queries a DNS name server for an address **nis** Uses Network Information Service protocol to obtain an address
alert	Checks addresses of remote sites attempting to access your system; you turn it on or off with the **on** and **off** options
nospoof	Confirms addresses of remote sites attempting to access your system
trim	Checks your local host's file; removes the domain name and checks only for the hostname; enables you to use only a hostname in your host file for an IP address
multi	Checks your local hosts file; allows a host to have several IP addresses; you turn it on or off with the **on** and **off** options

Table 32-7: Resolver Options, host.conf

In the next example of a **host.conf** file, the **order** option instructs your resolver first to look up names in your local **/etc/hosts** file, and then, if that fails, to query domain name servers. The system does not have multiple addresses.

/etc/host.conf

```
# host.conf file
# Lookup names in host file and then check DNS
order bind host
# There are no multiple addresses
multi off
```

/etc/nsswitch.conf: Name Service Switch

Different functions in the standard C Library must be configured to operate on your Linux system. Previously, database-like services, such as password support and name services like NIS or DNS, directly accessed these functions, using a fixed search order. For GNU C Library 2.*x,* used on current versions of Linux, this configuration is carried out by a scheme called the Name Service Switch (NSS), which is based on the method of the same name used by Sun Microsystems Solaris 2 OS. The database sources and their lookup order are listed in the **/etc/nsswitch.conf** file.

The **/etc/nsswitch.conf** file holds entries for the different configuration files that can be controlled by NSS. The system configuration files that NSS supports are listed in Table 32-8. An entry consists of two fields: the service and the configuration specification. The service consists of the configuration file followed by a colon. The second field is the configuration specification for that file, which holds instructions on how the lookup procedure will work. The configuration specification can contain service specifications and action items. Service specifications are the services to search. Currently, valid service specifications are nis, nis-plus, files, db, dns, and compat (see Table 32-9). Not all are valid for each configuration file. For example, the dns service is valid only for the **hosts** file, whereas nis is valid for all files. The following example will first check the local **/etc/password** file and then NIS.

```
passwd:  files nisplus
```

File	Description
aliases	Mail aliases, used by Sendmail
ethers	Ethernet numbers
group	Groups of users
hosts	Hostnames and numbers
netgroup	Network-wide list of hosts and users, used for access rules; C libraries before glibc 2.1 only support netgroups over NIS
network	Network names and numbers
passwd	User passwords
protocols	Network protocols
publickey	Public and secret keys for SecureRPC used by NFS and NIS+
rpc	Remote procedure call names and numbers
services	Network services
shadow	Shadow user passwords

Table 32-8: NSS-Supported Files

An action item specifies the action to take for a specific service. An action item is placed within brackets after a service. A configuration specification can list several services, each with its own action item. In the following example, the entry for the network file has a configuration specification that says to check the NIS service and, if not found, to check the **/etc/protocols** file:

```
protocols: nisplus [NOTFOUND=return] files
```

An action item consists of a status and an action. The status holds a possible result of a service lookup, and the action is the action to take if the status is true. Currently, the possible status values are SUCCESS, NOTFOUND, UNAVAIL, and TRYAGAIN (service temporarily unavailable). The possible actions are return and continue: return stops the lookup process for the configuration file, whereas continue continues on to the next listed service. In the preceding example, if the record is not found in NIS, the lookup process ends.

Service	Description
files	Checks corresponding **/etc** file for the configuration (for example, **/etc/hosts** for hosts); this service is valid for all files
db	Checks corresponding **/var/db** databases for the configuration; valid for all files except **netgroup**
compat	Valid only for **passwd**, **group**, and **shadow** files
dns	Checks the DNS service; valid only for **hosts** file
nis	Checks the NIS service; valid for all files
nisplus	NIS version 3
hesiod	Uses Hesiod for lookup

Table 32-9: NSS Configuration Services

Shown here is a copy of the **/etc/nsswitch.conf** file, which lists commonly used entries. Comments and commented-out entries begin with a **#** sign:

/etc/nsswitch.conf

```
#
# /etc/nsswitch.conf
#
# An example Name Service Switch config file.
passwd:                 db files nisplus nis
shadow:                db files nisplus nis
group:              db files nisplus nis
hosts:              files nisplus dns
bootparams:         nisplus [NOTFOUND=return] files
ethers:             files
netmasks:           files
networks:           files
protocols:          files
rpc:                files
services:           files
netgroup:           nisplus
publickey:          nisplus
automount:          files
aliases:           files nisplus
```

Network Interfaces and Routes: ifconfig and route

Your connection to a network is made by your system through a particular hardware interface, such as an Ethernet card or a modem. Data passing through this interface is then routed to your network. The **ifconfig** command configures your network interfaces, and the **route** command sets up network connections accordingly. If you configure an interface with a network configuration tool, such as system-config-network, you needn't use **ifconfig** or **route**. However, you can directly configure interfaces using **ifconfig** and **route**, if you want. Every time you start your system, the network interfaces and their routes must be established. This is done automatically for you when you boot up by **ifconfig** and **route** commands executed for each interface by the **/etc/rc.d/init.d/network** initialization file, which is executed whenever you start your system. If you are manually adding your own interfaces, you must set up the network script to perform the **ifconfig** and **route** operations for your new interfaces.

Network Startup Script: /etc/rc.d/init.d/network

Your network interface is started up using the **network** script in the **/etc/rc.d/init.d** directory. This script will activate your network interface cards (NICs) as well as implement configuration information such as gateway, host, and name server identities. You can manually shut down and start your network interface using this script and the **restart**, **start**, or **stop** options, as well as system-config-services. You can run the script with the **service** command. The following commands shut down and then start up your network interface:

```
service network stop
service network start
```

If you are changing network configuration, you will have to restart your network interface for the changes to take effect:

```
service network restart
```

To test if your interface is working, use the **ping** command with an IP address of a system on your network, such as your gateway machine. The **ping** command continually repeats until you stop it with a CTRL-C.

```
ping 192.168.0.1
```

Interface Configuration Scripts: /etc/sysconfig/network-scripts

The **/etc/rc.d/init.d/network** file performs the startup operations by executing several specialized scripts located in the **/etc/sysconfig/network-scripts** directory. The **network** script uses a script in that directory called **ifup** to activate a network connection, and **ifdown** to shut it down; **ifup** and **ifdown** will invoke other scripts tailored to the kind of device being worked on, such as **ifup-ppp** for modems using the PPP protocol, or **ifup-ipv6** for network devices that use IP Protocol version 6 addressing.

Note: You can activate and deactivate network interfaces using system-config-network accessible from the System | Administration | Network.

The **ifup** and **ifdown** scripts make use of interface configuration files that bear the names of the network interfaces currently configured, such as **ifcfg-eth0** for the first Ethernet device. These files define shell variables that hold information on the interface, such as whether to start

them at boot time. For example, the **ifcfg-eth0** file holds definitions for NETWORK, BROADCAST, and IPADDR, which are assigned the network, broadcast, and IP addresses that the device uses.

The **ifdown** and **ifup** scripts, in turn, hold the **ifconfig** and **route** commands to activate scripts using these variables defined in the interface configuration files. If you want to manually start up an interface with **ifup**, you simply use the interface configuration file as its argument. The following command starts up the second Ethernet card:

```
cd /etc/sysconfig/network-scripts
ifup ifcfg-eth1
```

Interface configuration files are automatically generated when you configure your network connections, such as with a Red Hat and Fedora's system-config-network administrative tool . You can also manually edit these interface configuration files, making changes such as whether to start up the interface at boot or not (though using a configuration tool such as system-config-network is easier). A sample **ifcfg-eth0** file is shown here using a static IP address:

/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.0.255
IPADDR=192.168.0.1
NETMASK=255.255.255.0
NETWORK=192.168.0.0
ONBOOT=yes
```

A DHCP-based interface would look something like this, where BOOTPROTO is assigned dhcp:

/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
BOOTPROTO=dhcp
HWADDR=00:00:00:00:00:01
TYPE=Ethernet
ONBOOT=yes
```

ifconfig

The **ifconfig** command takes as its arguments the name of an interface and an IP address, as well as options. The **ifconfig** command then assigns the IP address to the interface. Your system now knows that such an interface exists and that it references a particular IP address. In addition, you can specify whether the IP address is a host address or a network address. You can use a domain name for the IP address, provided the domain name is listed along with its IP address in the **/etc/hosts** file. The syntax for the **ifconfig** command is as follows:

```
# ifconfig interface -host_net_flag address options
```

The *host_net_flag* can be either **-host** or **-net** to indicate a host or network IP address. The **-host** flag is the default. The **ifconfig** command can have several options, which set different features of the interface, such as the maximum number of bytes it can transfer (**mtu**) or the

broadcast address. The **up** and **down** options activate and deactivate the interface. In the next example, the **ifconfig** command configures an Ethernet interface:

```
ifconfig eth0 192.168.0.1
```

ifconfig lo

Option	Description
Interface	Name of the network interface, such as **eth0** for the first Ethernet device or **ppp0** for the first PPP device (modem)
up	Activates an interface; implied if IP address is specified
down	Deactivates an interface
allmulti	Turns on or off the promiscuous mode; preceding hyphen (**-**) turns it off; this allows network monitoring
mtu *n*	Maximum number of bytes that can be sent on this interface per transmission
dstaddr *address*	Destination IP address on a point-to-point connection
netmask *address*	IP network mask; preceding hyphen (**-**) turns it off
broadcast *address*	Broadcast address; preceding hyphen (**-**) turns it off
point-to-point *address*	Point-to-point mode for interface; if address is included, it is assigned to remote system
hw	Sets hardware address of interface
Address	IP address assigned to interface

Table 32-10: The ifconfig Options

For a simple configuration such as this, **ifconfig** automatically generates a standard broadcast address and netmask. The standard broadcast address is the network address with the number 255 for the host address. For a class C network, the standard netmask is 255.255.255.0, whereas for a class A network, the standard netmask is 255.0.0.0. If you are connected to a network with a particular netmask and broadcast address, however, you must specify them when you use **ifconfig**. The option for specifying the broadcast address is **broadcast**; for the network mask, it is **netmask**. Table 32-10 lists the different **ifconfig** options. In the next example, **ifconfig** includes the netmask and broadcast address:

```
ifconfig eth0 192.168.0.1 broadcast 192.168.0.255 netmask 255.255.255.0
```

Once you configure your interface, you can use **ifconfig** with the **up** option to activate it and with the **down** option to deactivate it. If you specify an IP address in an **ifconfig** operation, as in the preceding example, the **up** option is implied.

```
ifconfig eth0 up
```

Point-to-point interfaces such as Parallel IP (PLIP), Serial Line IP (SLIP), and Point-to-Point Protocol (PPP) require you to include the **pointopoint** option. A PLIP interface name is identified with the name **plip** with an attached number. For example, **plip0** is the first PLIP interface. SLIP interfaces use **slip0**. PPP interfaces start with **ppp0**. Point-to-point interfaces are

those that usually operate between only two hosts, such as two computers connected over a modem. When you specify the **pointopoint** option, you need to include the IP address of the host. In the next example, a PLIP interface is configured that connects the computer at IP address 192.168.1.72 with one at 204.166.254.14. If domain addresses were listed for these systems in **/etc/hosts**, those domain names could be used in place of the IP addresses.

```
ifconfig plip0 192.168.1.72 pointopoint 204.166.254.14
```

If you need to, you can also use **ifconfig** to configure your loopback device. The name of the loopback device is **lo**, and its IP address is the special address 127.0.0.1. The following example shows the configuration:

```
ifconfig lo 127.0.0.1
```

The **ifconfig** command is useful for checking on the status of an interface. If you enter the **ifconfig** command along with the name of the interface, information about that interface is displayed:

```
ifconfig eth0
```

To see if your loopback interface is configured, you can use **ifconfig** with the loopback interface name, **lo**:

Routing

A packet that is part of a transmission takes a certain *route* to reach its destination. On a large network, packets are transmitted from one computer to another until the destination computer is reached. The route determines where the process starts and to what computer your system needs to send the packet for it to reach its destination. On small networks, routing may be static—that is, the route from one system to another is fixed. One system knows how to reach another, moving through fixed paths. On larger networks and on the Internet, however, routing is dynamic. Your system knows the first computer to send its packet off to, and then that computer takes the packet from there, passing it on to another computer, which then determines where to pass it on. For dynamic routing, your system needs to know little. Static routing, however, can become complex because you have to keep track of all the network connections.

Your routes are listed in your routing table in the **/proc/net/route** file. To display the routing table, enter **route** with no arguments (the **netstat -r** command will also display the routing table):

```
# route
Kernel routing table
Destination Gateway      Genmask      Flags Metric Ref Use  Iface
192.168.0.0    *         255.255.255.0  U     0      0  0    etho
127.0.0.0      *         255.0.2055.0   U     0      0  0    lo
default     192.168.0.1  0.0.0.0        UG    0      0  0    eth0
```

Each entry in the routing table has several fields, providing information such as the route destination and the type of interface used. The different fields are listed in Table 32-11.

Field	Description
Destination	Destination IP address of the route
Gateway	IP address or hostname of the gateway the route uses; * indicates no gateway is used
Genmask	The netmask for the route
Flags	Type of route: U = up, H = host, G = gateway, D = dynamic, M = modified
Metric	Metric cost of route
Ref	Number of routes that depend on this one
Window	TCP window for AX.25 networks
Use	Number of times used
Iface	Type of interface this route uses

Table 32-11: Routing Table Entries

With the **add** argument, you can add routes either for networks with the **-net** option or with the **-host** option for IP interfaces (hosts). The **-host** option is the default. In addition, you can then specify several parameters for information, such as the netmask (**netmask**), the gateway (**gw**), the interface device (**dev**), and the default route (**default**). If you have more than one IP interface on your system, such as several Ethernet cards, you must specify the name of the interface using the **dev** parameter. If your network has a gateway host, you use the **gw** parameter to specify it. If your system is connected to a network, at least one entry should be in your routing table that specifies the default route. This is the route taken by a message packet when no other route entry leads to its destination. The following example is the routing of an Ethernet interface:

```
route add 192.168.1.2 dev eth0
```

If your system has only the single Ethernet device as your IP interface, you could leave out the **dev eth0** parameter:

```
route add 192.168.1.2
```

You can delete any route you establish by invoking **ifconfig** with the **del** argument and the IP address of that route, as in this example:

```
route del 192.168.1.2
```

For a gateway, you first add a route to the gateway interface, and then add a route specifying that it is a gateway. The address of the gateway interface in this example is 192.168.1.1:

```
route add 192.168.1.1
route add default gw 192.168.1.1
```

If you are using the gateway to access a subnet, add the network address for that network (in this example, 192.168.23.0):

```
route add -net 192.168.23.0 gw dev eth1
```

To add another IP address to a different network interface on your system, use the **ifconfig** and **route** commands with the new IP address. The following command configures a second Ethernet card (**eth1**) with the IP address 192.168.1.3:

```
ifconfig eth1 192.168.1.3
route add 192.168.1.3 dev eth1
```

Monitoring Your Network: ping, netstat, tcpdump, EtherApe, Ettercap, and Wireshark

Several applications are available on Linux to let you monitor your network activity. Graphical applications like EtherApe, Ettercap, and Wireshark provide detailed displays and logs to let you analyze and detect network usage patterns. Other tools like **ping**, **netstat**, and **traceroute** offer specific services (see Table 32-12).

The EtherApe, Ettercap, and Wireshark tools can be accessed on the Applications | Internet menu. Tools like **ping**, **traceroute**, and **netstat** can be accessed with the Gnome Network Tools application accessible on the Applications | System Tools menu, as well as being rung individually on a command line (Terminal window).

EtherApe provides a simple graphical display for your protocol activity. The Preferences dialog lets you set features like the protocol to check and the kind of traffic to report.

Network Information Tools	Description
ping	Detects whether a system is connected to the network.
finger	Obtains information about users on the network.
who	Checks what users are currently online.
whois	Obtains domain information.
host	Obtains network address information about a remote host.
traceroute	Tracks the sequence of computer networks and hosts your message passes through.
wireshark	Protocol analyzer to examine network traffic.
gnome-nettool	GNOME interface for various network tools including ping, finger, and traceroute.
mtr and **xmtr**	My traceroute combines both ping and traceroute operations (Traceroute on System Tools menu).
EtherApe	Analyze protocol activity
Ettercap	Sniffer program for man-in-middle attacks
netstat	Real time network status monitor
tcpdump	Capture and save network packets

Table 32-12: Network Tools

GNOME Network Tools: gnome-nettool

For the GNOME desktop, the **gnome-nettool** utility provides a GNOME interface for entering the **ping** and **traceroute** commands (see Figure 32-4), as well as Finger, Whois, and

Lookup for querying users and hosts on the network (Applications | System Tools | Network Tools). Whois will provide domain name information about a particular domain, and Lookup will provide both domain name and IP addresses. It also includes network status tools such as **netstat** and **portscan**. The first tab, Devices, describes your connected network devices, including configuration and transmission information about each device, such as the hardware address and bytes transmitted. Both IPv4 and IPv6 host IP addresses will be listed.

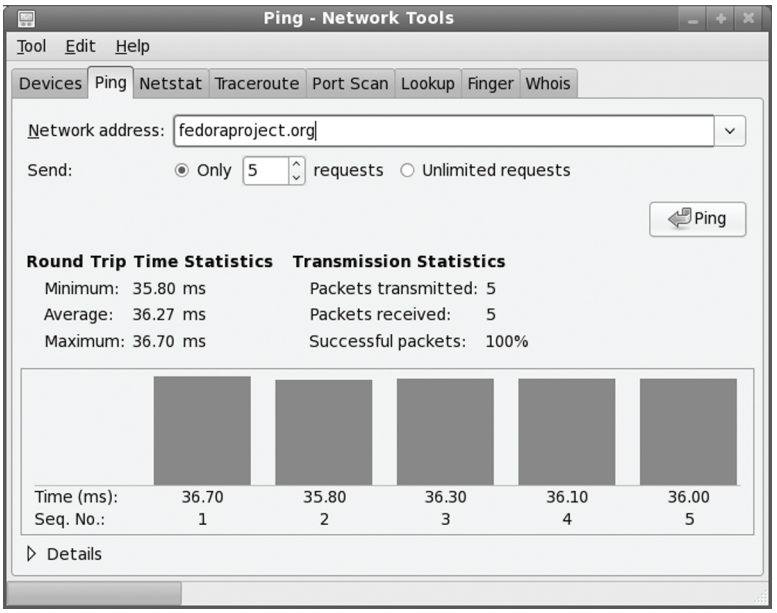


Figure 32-4: Gnome network tool

Network Information: ping, finger, traceroute, and host

You can use the **ping**, **finger**, **traceroute**, and **host** commands to find out status information about systems and users on your network. The **ping** command is used to check if a remote system is up and running. You use **finger** to find out information about other users on your network, seeing if they are logged in or if they have received mail; **host** displays address information about a system on your network, giving you a system's IP and domain name addresses; and **traceroute** can be used to track the sequence of computer networks and systems your message passed through on its way to you. Table 32-12 lists various network information tools.

ping

The **ping** command detects whether a system is up and running. **ping** takes as its argument the name of the system you want to check. If the system you want to check is down, **ping** issues a timeout message indicating a connection could not be made. The next example checks to see if **www.redhat.com** is up and connected to the network:

```
# ping www.redhat.com
PING www.redhat.com (209.132.177.50) 56(84) bytes of data.
64 bytes from www.redhat.com (209.132.177.50):icmp_seq=1 ttl=118 time=36.7ms
64 bytes from www.redhat.com (209.132.177.50):icmp_seq=2 ttl=118 time=36.9ms
--- www.redhat.com ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3000ms
rtt min/avg/max/mdev = 36.752/37.046/37.476/0.348 ms
```

You can also use **ping** with an IP address instead of a domain name. With an IP address, **ping** can try to detect the remote system directly without having to go through a domain name server to translate the domain name to an IP address. This can be helpful for situations where your network's domain name server may be temporarily down and you want to check if a particular remote host on your network is connected.

```
# ping 209.132.177.50
```

Note: A **ping** operation could also fail if **ping** access is denied by a network's firewall.

finger and who

You can use the **finger** command to obtain information about other users on your network and the **who** command to see what users are currently online on your system. The **who** and **w** commands lists all users currently connected, along with when, how long, and where they logged in. The **w** command provides more detailed information. It has several options for specifying the level of detail. The **who** command is meant to operate on a local system or network; **finger** can operate on large networks, including the Internet, though most systems block it for security reasons.

Note: Wireshark is a protocol analyzer that can capture network packets and display detailed information about them. You can detect what kind of information is being transmitted on your network as well as its source and destination. Wireshark is used primarily for network and server administration.

host

With the **host** command, you can find network address information about a remote system connected to your network. This information usually consists of a system's IP address, domain name address, domain name nicknames, and mail server. This information is obtained from your network's domain name server. For the Internet, this includes all systems you can connect to over the Internet.

The **host** command is an effective way to determine a remote site's IP address or URL. If you have only the IP address of a site, you can use **host** to find out its domain name. For network administration, an IP address can be helpful for making your own domain name entries in your **/etc/host** file. That way, you needn't rely on a remote domain name server (DNS) for locating a site.

```
# host gnomefiles.org
gnomefiles.org has address 67.18.254.188
gnomefiles.org mail is handled by 10 mx.zayda.net.

# host 67.18.254.188
188.254.18.67.in-addr.arpa domain name pointer gnomefiles.org.
```

traceroute

Internet connections are made through various routes, traveling through a series of interconnected gateway hosts. The path from one system to another could take different routes, some of which may be faster than others. For a slow connection, you can use **traceroute** to check the route through which you are connected to a host, monitoring the speed and the number of intervening gateway connections a route takes. The **traceroute** command takes as its argument the hostname or IP addresses for the system whose route you want to check. Options are available for specifying parameters like the type of service (**-t**) or the source host (**-s**). The **traceroute** command will return a list of hosts the route traverses, along with the times for three probes sent to each gateway. Times greater than five seconds are displayed with a asterisk, *.

```
traceroute rabbit.mytrek.com
```

You can also use the mtr or xmtr tools to perform both ping and traces (Traceroute on the System Tools menu).

Ettercap

Ettercap is a sniffer program designed to detect Man in the Middle attacks. In this kind of attack, packets are detected and modified in transit to let an unauthorized user access a network. You can use either its graphical interface or its command line interface. Ettercap can perform Unified sniffing on all connections, or Bridged sniffing on a connection between network interfaces. Ettercap uses plugins for specific tasks, like dos_attack to detect Denial of Service attacks and dns-spoof for DNS spoofing detection. Check the plugins Help panel, or enter **ettercap -P list** for a complete listing. Ettercap can be run in several modes, including a text mode, a command line cursor mode, a script mode using commands in a file, and even as a daemon logging results automatically.

Wireshark

Wireshark is a network protocol analyzer that lets you capture packets transmitted across your network, selecting and examining those from protocols you want to check. You can examine packets from a particular transmissions, displaying the data in readable formats. The Wireshark interface displays three panes: a listing of current packets, the protocol tree for the currently selected packet, a display of the selected packets contents. The first pane categorizes entries by time, source, destination, and protocol. There are button headers for each. To sort a set of entries by a particular category, click its header. For example, group entries by protocol, click the Protocol button; for destinations, use the Destination button.

Capture Options

To configure Wireshark, you select the Options entry from the Capture menu. This opens an options window where you can select the network interface to watch. Here you can also select options such as the file to hold your captured information in and a size limit for the capture, along with a filter to screen packets. With the promiscuous mode selected, you can see all network traffic passing through that device, whereas with it off, you will see only those packets destined for that device. You can then click the start button to start Wireshark. To stop and start Wireshark, you select the Stop and Start entries on the Capture menu.

- The Capture Files option lets you select a file to save your capture in. If no file is selected, then data is simply displayed in the Wireshark window. If you want to keep a continuous running snapshot of your network traffic, you can use ring buffers. These are a series of files that are used to save captured data. When they fill up, the capture begins saving again to the first file, and so on. Check Use multiple file to enable this option.
- Display options control whether packets are displayed in real time on the Wireshark window.
- Limits let you set a limit for the capture packet size.
- Capture filter lets you choose the type of protocol you want to check.
- Name resolution enables the display of host and domain names instead of IP addresses, if possible.

Wireshark Filters

A filter lets you select packets that match specified criteria, such as packets from a particular host. Criteria are specified using expressions supported by the Packet Capture Library and implemented by **tcpdump**. Wireshark filters use similar expressions as those used by the **tcpdump** command. Check the **tcpdump** Man page for detailed descriptions.

You can set up either a Search filter in the Find tab (Edit menu) to search for certain packets, or set up a Capture filter in the Options tab (Capture menu) to select which packets to record. The filter window is the same for both. On the filter window you can select the protocol you want to search or capture. The Filter name and string will appear in the Properties segment. You can also enter your own string, setting up a new filter of your own. The string must be a filter expression.

To create a new filter, enter the name you want to give it in the Filter Name box. Then in the Filter String box, enter the filter expression, like **icmp**. Then click New. Your new filter will appear in the list. To change a filter, select it and change its expression in the Filter String box, then click Change.

A filter expression consists of an ID, such as the name or number of host, and a qualifier. Qualifiers come in three types: type, direction, and protocol. The type can reference the host, network, or port. The type qualifiers are **host**, **net**, and **port**. Direction selects either source or destination packets, or both. The source qualifier is **src**, and the destination, **dst**. With no destination qualifier, both directions are selected. Protocol lets you specify packets for a certain protocol. Protocols are represented using their lowercase names, such as **icmp** for ICMP. For example, the expression to list all packets coming in from a particular host would be **src host** *hostname,* where *hostname* is the source host. The following example will display all packets from the 192.168.0.3 host:

```
src host 192.168.0.3
```

Using just **host** will check for all packets going out as well as coming in for that host. The **port** qualifier will check for packets passing through a particular port. To check for a particular protocol, you use the protocol name. For example, to check for all ICMP packets you would use the expression

```
icmp
```

There are also several special qualifiers that let you further control your selection. The **gateway** qualifier lets you detect packets passing through a gateway. The **broadcast** and **multicast** qualifiers detect packets broadcast to a network. The **greater** and **less** qualifiers can be applied to numbers such as ports or IP addresses.

You can combine expressions into a single complex Boolean expression using **and**, **or**, or **not**. This lets you create a more refined filter. For example, to capture only the ICMP packets coming in from host 192.168.0.2, you can use

```
src host 192.168.0.3 and icmp
```

tcpdump

Like Wireshark, **tcpdump** will capture network packets, saving them in a file where you can examine them. **tcpdump** operates entirely from the command line. You will have to open a terminal window to run it. Using various options, you can refine your capture, specifying the kinds of packets you want. **tcpdump** uses a set of options to specify actions you want to take, which include limiting the size of the capture, deciding which file to save it to, and choosing any filter you want to apply to it. Check the **tcpdump** Man page for a complete listing.

- The **-i** option lets you specify an interface to listen to.
- With the **-c** option, you can limit the number of packets to capture.
- Packets will be output to the standard output by default. To save them to a file, you can use the **-w** option.
- You can later read a packet file using the **-r** option and apply a filter expression to it.

The **tcpdump** command takes as its argument a filter expression that you can use to refine your capture. Wireshark uses the same filter expressions as **tcpdump** (see the filters discussion in Wireshark).

netstat

The netstat program provides real-time information on the status of your network connections, as well as network statistics and the routing table. The **netstat** command has several options you can use to bring up different sorts of information about your network.

```
# netstat
Active Internet connections
Proto Recv-Q Send-Q Local Address Foreign Address (State) User
tcp 0 0 turtle.mytrek.com:01 pango1.mytrain.com.:ftp ESTABLISHED dylan
Active UNIX domain sockets
Proto RefCnt Flags Type State Path
unix 1 [ ACC ] SOCK_STREAM LISTENING /dev/printer
unix 2 [ ] SOCK_STREAM CONNECTED /dev/log
unix 1 [ ACC ] SOCK_STREAM LISTENING /dev/nwapi
unix 2 [ ] SOCK_STREAM CONNECTED /dev/log
unix 2 [ ] SOCK_STREAM CONNECTED
unix 1 [ ACC ] SOCK_STREAM LISTENING /dev/log
```

The **netstat** command with no options lists the network connections on your system. First, active TCP connections are listed, and then the active domain sockets are listed. The domain

sockets contain processes used to set up communications among your system and other systems. You can use **netstat** with the **-r** option to display the routing table, and **netstat** with the **-i** option displays the uses of the different network interfaces.

IP Aliasing

In some cases, you may want to assign a single Linux system that has only one network interface to two or more IP addresses. For example, you may want to run different Web sites that can be accessed with separate IP addresses on this same system. In effect, you are setting up an alias for your system, another address by which it can be accessed. In fact, you are assigning two IP addresses to the same network interface—for example, assigning a single Ethernet card two IP addresses. This procedure, referred to as *IP aliasing,* is used to set up multiple IP-based virtual hosts for Internet servers. This method enables you to run several Web servers on the same machine using a single interface (or more than one on each of several interfaces). See Chapters 21 and 22 for FTP and Web server information about virtual hosts, and Chapter 30 for Domain Name Service configuration.

Setting up an IP alias is a simple matter of configuring a network interface on your system to listen for the added IP address. Your system needs to know what IP addresses it should listen for and on what network interface. You set up IP aliases using the **ifconfig** and **route** commands, or a network administrative tool.

To add another address to the same interface, you need to qualify the interface by adding a colon and a number. For example, if you are adding another IP address to the first Ethernet card (**eth0**), you would add a **:0** to its interface name, **eth0:0**. The following example shows the **ifconfig** and **route** commands for the Ethernet interface 192.168.1.2 and two IP aliases added to it: 192.168.1.100 and 192.168.1.101. To add yet another IP address to this same interface, you would use **eth0:1**, incrementing the qualifier, and so on. The first **ifconfig** command assigns the main IP address, 192.168.1.2, to the first Ethernet device, **eth0**. Then, two other IP addresses are assigned to that same device. In the first **route** command, the network route is set up for the Ethernet device, and then routes are set up for each IP interface. The interfaces for the two aliases are indicated with **eth0:0** and **eth0:1**:

```
ifconfig eth0 192.168.1.2
ifconfig eth0:0 192.168.1.100
ifconfig eth0:1 192.168.1.101
route add -net 192.168.1.0 dev eth0
route add -host 192.168.1.2 dev eth0
route add -host 192.168.1.100 dev eth0:0
route add -host 192.168.1.101 dev eth0:1
```

IP aliasing must be supported by the kernel before you can use it. If your kernel does not support it, you may have to rebuild the kernel (including IP aliasing support), or use loadable modules to add IP aliasing.

Tip: InfiniBand is often used as a replacement for local network connections. Check the Linux InfiniBand Project **http://sourceforge.net/projects/infiniband**.

Appendix A: Getting Fedora

The Fedora Linux distribution installs a professional-level and very stable Linux system along with the KDE and GNOME GUI interfaces, flexible and easy-to-use system configuration tools, an extensive set of Internet servers, a variety of different multimedia applications, and thousands of Linux applications of all kinds. You can find recent information about Red Hat Fedora Project at **www.fedoraproject.org**.

Most Fedora software is available for download from the Fedora repository. Install disks are available also, either as smaller desktop only installs, or larger server installs. You can every use the Live CD to install Fedora. Fedora distribution strategy relies on install disks with a selected collection of software that can be later updated and enhanced from the very large collection of software on the Fedora repository. This means that the collection of software in an initial installation can be relatively small. Software on the Fedora repository is also continually updated, so any installation will likely have to undergo extensive updates from the repository.

Downloading Disc Images

To obtain Fedora 10, just go to the Fedora Project Web site and click on the Get Fedora link. Here you will see links for both bittorrent or direct download for the different Fedora Install DVD and Fedora Live versions. A detailed description for all the Fedora download options, including all the ISO discs you will need and links to mirror sites, is displayed. The Direct downloads link to a currently available mirror site.

`http://fedoraproject.org/get-fedora`

You could also directly access a Fedora mirror site by entering the following URL. You would then need to navigate through the **releases** and 10 directories to find the Fedora and Live directories where the Fedora Install and Live iso images are kept.

`http://download.fedoraproject.org`

You can also access a specific mirror at the following URL. Here will be listed the current Fedora mirror and their addresses.

`http://mirrors.fedoraproject.org`

at the following URL. Check this site for the latest download procedures.

Alternativley, you can download the Everything spin from the Fedora Unity Project, as well as periodic re-spins of Fedora ISO images with updated software packages.

http://spins.fedoraunity.org

You can also still download directly from the Fedora site using the following link at **http://download.fedora.redhat.com**.

http://download.fedora.redhat.com/pub/fedora/linux/releases/

The direct downloads can be very fast with broadband DSL or cable, using an FTP client like **gFTP** (Applications | Internet menu). Web-client download with browsers like Firefox tend to be slower.

Install Install and Live Images

To download Fedora 10 for installation from a DVD/CD-ROM drive, you download either the Fedora Install DVD image or a Fedora Live image. The Fedora Install and Live images are large files that have the extension .iso. The Fedora Install DVD resides within the Fedora subdirectory, under the respective version (i386, x86_64, or ppc), in an iso directory (Fedora/i386/iso). The Live images reside in the Live subdirectory, under the respective versions (i686, x86_64, and ppc). Once they are downloaded, you burn them to a disc using your CD or DVD writer and burner software, like the Nautilus file browser or K3b on Fedora.

There are ISO images for 64-bit system support and for the standard x86 (32-bit) support. Download the appropriate one. You cannot run a 64-bit version on an x86 (32-bit) system.

You do not have to download the images to a Linux system. You can just as easily download them on a Windows system and use Windows CD/DVD burner software to make the discs.

Using BitTorrent

You can use any FTP or Web client, such as gFTP or Firefox, to download the CD image files. The DVD image, though, is a very large file that can take a long time to download, especially if the FTP site is very busy or if your have a slow Internet connection. An alternative for such very large files is to use BitTorrent. BitTorrent is a safe distributed download operation that is ideal for large files, letting many participants download and upload the same file, building a torrent that can run very fast for all participants. The Fedora 10 BitTorrent files are located both at **www.fedoraproject.org** and at

http://torrent.fedoraproject.org/

You will first need to install the BitTorrent client. For Fedora, there are several BitTorrent clients available, including **azureus**, **rtorrent**, **ktorrent**, and the original **bittorrent**. A search on "bittorrent" in Add/Remove Software will display them (notice that bittorrent has two ‘t’s and two ‘r’s in its spelling). Just install the ones you want.

Appendix B: Fedora Live DVD/CD

Fedora Live DVD/CD

With the Fedora Live DVD/CD you can run Fedora from any DVD/CD-ROM drive. Some images use CDs and others use DVDs. In effect, you can carry your operating system with you on just a DVD/CD-ROM. New users could also use the Live-DVD/CD to check out Fedora to see if they like it. Files and data can be written to removable devices like USB drives, but the OS configuration cannot be changed. You could also mount partitions from hard drives on the system you are running the Live DVD/CD on. You can find out more about the Fedora Live DVD/CD at **www.fedoraproject.org/wiki/FedoraLiveCD**.

The Live DVD/CD provided by Fedora includes a very limited set of software packages. OpenOffice applications are not provided, just Abiword for word processing and Gnumeric for spread sheets. Servers are not included. For desktop support you use GNOME. Other than these limitations, you have a fully operational Fedora desktop. You have the full set of administrative tools, with which you can add users and change configuration settings while the Live DVD/CD is running. When you shut down, the configuration information is lost.

You can logout, but this will display a login screen with a simple login button and a language menu. You can use the language menu to change your language. But you cannot shut down or restart from the login menu. You can only do that from the desktop.

The Live DVD/CD enables NetworkManager by default, automatically detecting and configuring your network connection. The Live DVD/CDs have a default username that has no password.

Fedora provides both official and custom spins. The Live CD images are available from **http://fedoraproject.org/get-fedora** page, on the LiveCD segment. Fedora provides a GNOME and KDE desktop official Live CD spins.

Fedora Live: Available for i868, x86_64, and ppc. Includes Gnome desktop and productivity applications. 686 is a Live CD, x86_64 and ppc are DVDs.

Fedora KDE Live: Available for i686 and x86_64. Includes KDE desktop. 686 version is a CD, x86_64 is a DVD.

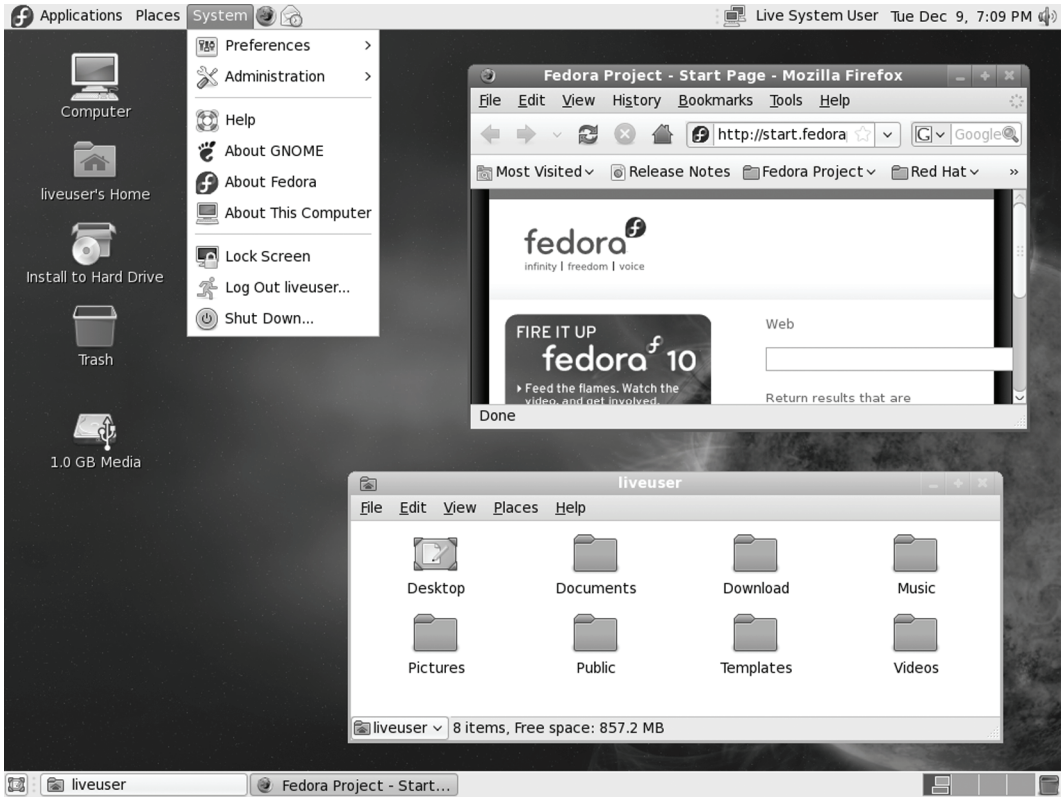


Figure B-4: Fedora Live CD

Starting the Live DVD/CD

When you first boot, you can press the spacebar to display the boot options. These include: booting up the Live DVD/CD directly (default), verifying the disk media first, performing a memory test, or to boot another OS already on your hard disk. If you press nothing, then the Live DVD/CD starts up automatically. After your system starts up, you will be presented with the standard Login screen (the KDE disk will boot directly to the desktop). You will be automatically logged in. The GNOME (see Figure B-1) or KDE (see Figure B-2) desktop will then start up.

An easy way to save data from a Live session is to use a USB drive. On GNOME, the USB drive will appear as an icon on the desktop. Double click on it to open a file manager window for the USB drive. You can then copy files generated during the session to the USB drive. Remember to eject the drive before removing (right-click on icon and select Eject).

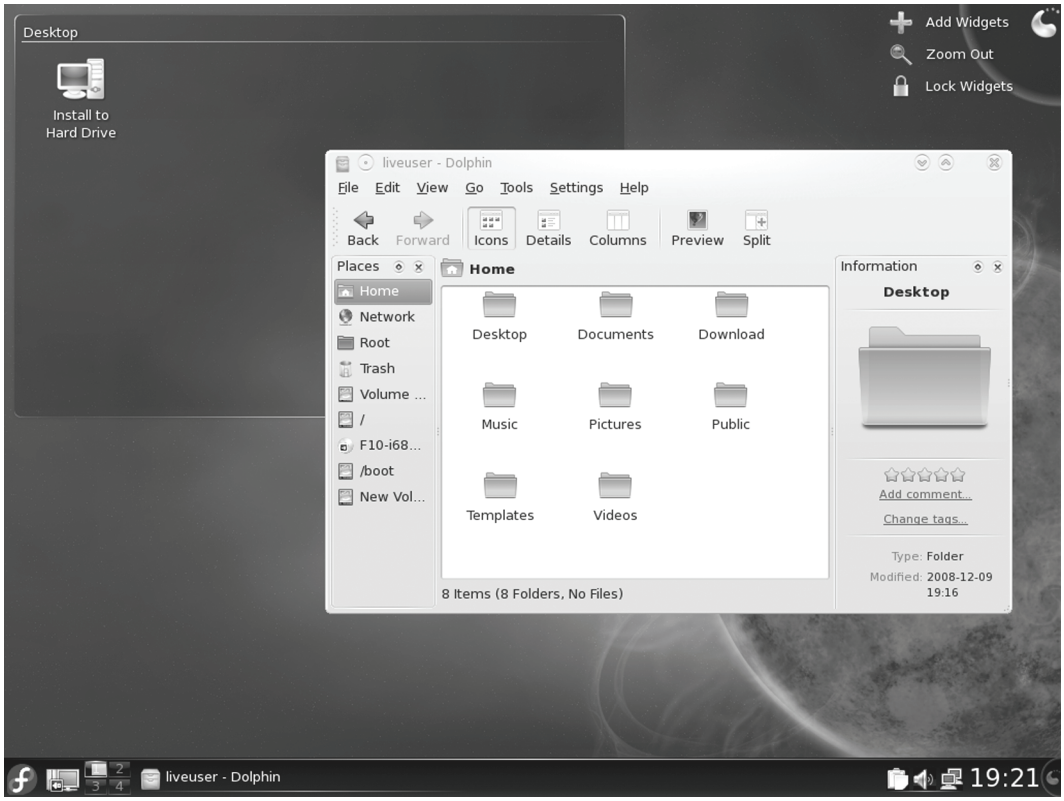


Figure B-5: Fedora KDE Live CD

Using USB drives on KDE

The KDE Live CD uses the new Plasma desktop and panel. USB drives are no longer displayed as icons on the desktop. An icon for the USB drive will appear on the Kickoff Computer menu, in the file manager sidebar, and the New Device Notifier applet (computer screen image next to Fedora menu image on the left side of the panel). To mount the drive, just click its entry in any of these locations, usually the New Device Notifier. A mount emblem will appear on the drive (see Figure B-3). On the file manager window, click the USB drive entry in the sidebar to open the USB drive file listing. You can then save files to the drive.

To remove a USB drive using either the Kickoff menu or the New Device Notifier, pass your mouse over the USB entry in either. An Eject button will appear. Click on that button to unmount the USB drive. You can then remove it. On the file manager sidebar entry, right-click on the USB entry. This opens a menu with an option to "Safely remove Volume". Select this option to finish writing any data to the drive. The mount emblem will disappear from the USB icon and you can then safely remove the USB drive.

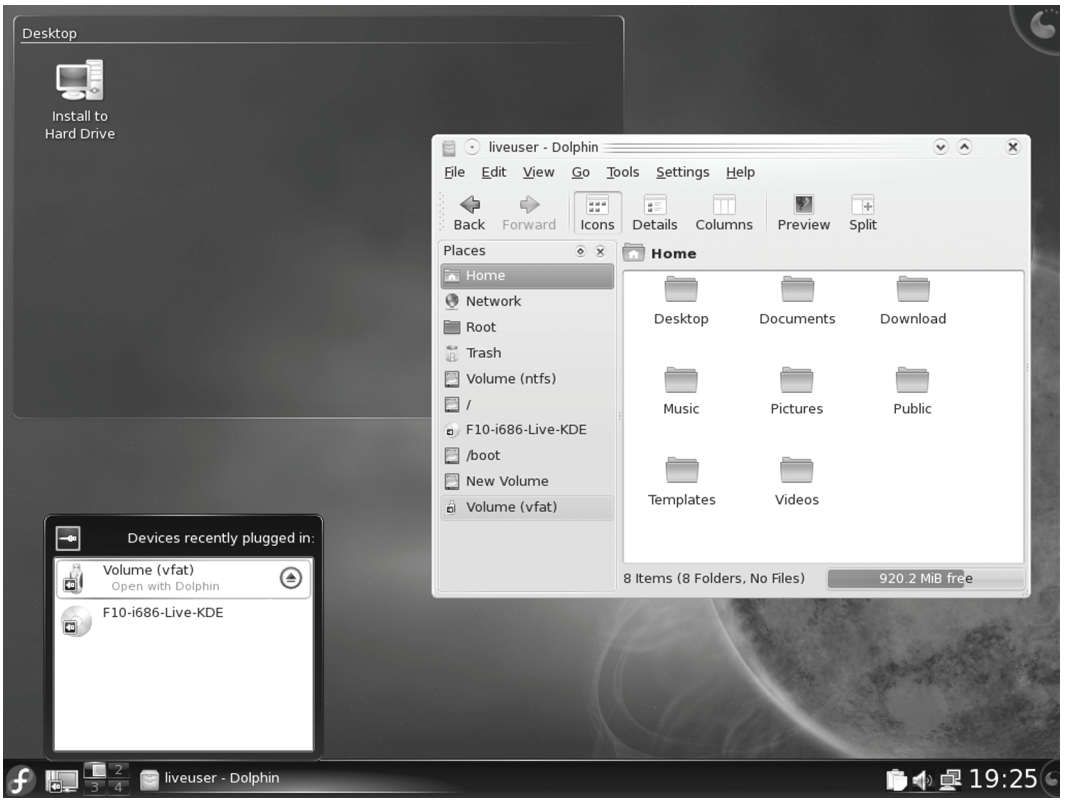


Figure B-3: Fedora KDE Live CD USB drive as shown on the New Device Notifier (lower left) and the file manager sidebar (last entry).

Installing Fedora from a Live CD

The Live CD can also be used as an installation disk, providing its limited collection of software on a system, but installing a full fledged Fedora operating system that can be expanded and updated from Fedora online repositories (see Chapter 4). In this case you would not have to download a complete set of Fedora installation disks, just the Live CD, and add packages later from repositories. Double-click on the "Install to Hard Drive" icon on the desktop to start the installation. Installs from a Live disk will enable NetworkManager by default and disable the SSH daemon (sshd).

Create your own Live CD

You can also create your own Live CD which you can have include your favorite software. From an installed Fedora system, you can use the **livecd-creator** to create your own Live CD (livecd-tools package). Install the **livecd-tools** package, if not already installed. The **livecd-creator** tool uses a configuration file set up in kickstart syntax to create a Live CD ISO image. Live CD kickstart configuration files for a minimal, desktop (GNOME), and KDE package selection is located at **/usr/share/livecdtools**.

The livecd-creator help option provides a complete listing of options with examples, **--help**. You use the **--config** option to specify a kickstart configuration file, the **--label** option to name your disk, and the **--repo** option to specify any special repositories. Check the README file in /usr/shard/doc/livecd-tools version for detailed information. The livecd-creator tool will first create a disk image, then download specified packages and install them on the image, install the boot loader, and then create the ISO image. The image and packages are held in the /var/tmp directory. The following creates a simple GNOME desktop Live CD (entered as one line).

```
livecd-creator --config=/usr/share/livecd-tools/livecd-fedora-desktop.ks --
fslabel=MyFedora-Live
```

In the configuration file you set start up options like language, firewall, xconfig for graphical start up, and specific services like NetworkManager and DHCP for network connections. The repositories used are specified in the **repo** entries. You can use the desktop, minimal, and KDE configuration files as a base to work from. Packages are listed after the **%packages** entry. For individual packages you simply specify the package's unique name. Kickstart syntax for groups of packages conforms to the same categories and subheadings use in Add/Remove Software category sidebar panel (PackageKit). These are useful when you have to select an extensive set packages like those used for the GNOME or KDE desktops. Be sure to prefix a group name with the **@** symbol, like **@Base** for the base packages. .

USB Live Disk

You can also install Fedora Live images to a USB disk. The procedure is not destructive. Your original data on the USB disk is preserved. To create a Live USB drive, you can either use the liveusb-creator application (**liveusb-creator** package), or the livecd-iso-to-disk command (**livecd-tools** package).

liveusb-creator

The liveusb-creator application is a GNOME application with an easy to use interface for creating a Live USB image from a Live CD iso image file. Once installed you can start liveusb-creator from Applications | System Tools | liveusb-creator. This opens the Fedora LiveUSB Creator window as shown in Figure B-4. Use the Browse button to locate the USB image or download one using the Download Fedora pop-up menu. The selected image will appear in the pane below. Use the Target Device pop-up menu to select the USB drive to use, if more than one. The Persistent Storage slider allows you to create an overlay memory segment on which changes and added data can be saved. When you are ready, just click the Create Live USB button on the bottom of the window.

livecd-iso-to-disk

You can also install Fedora Live images to a USB disk using the **livecd-iso-to-disk** command to install the image (part of the **livecd-tools** package). This is a command line tool that you enter in a Terminal window. Use the Live image and the device name the USB disk as your arguments.

```
/usr/bin/livecd-iso-to-disk    F10-i686-Live.iso    /dev/sdb1
```

Each Live CD also provides a **livecd-iso-to-disk** script in its LiveOS directory.

Figure B-4: Fedora LiveUSB Creator

Live USB Persistence Storage

If you want to be able to make changes to the Fedora OS on the USB Live version, you set up an overlay memory segment on the USB drive. To do this, use the **--overlay-size-mb** option with the size of the overlay in megabytes. Be sure your USB drive is large enough to accommodate both the overlay memory and the CD image. The following allows for 512 MB of persistent data that will be encrypted.

```
livecd-iso-to-disk --overlay-size-mb 512    F10-i686-Live.iso  /dev/sbd1
```

Persistent Home Directory

If you want to be able to save data to a **/home** directory on the USB Live version, you set up a home directory memory segment. To do this, use the **--home-size-mb** option with the size of the home directory segment in megabytes. Be sure your USB drive is large enough to accommodate the home memory and the CD image, as well as a memory overly if you also want to enable changes to the operating system. Your **/home** directory memory segment will be encrypted by default to protect your data in case it is lost or stolen. Upon creating your overlay, you will be prompted for a passphrase. Whenever to boot up your USB system, you will be prompted for the passphrase. The following allows for 1024MB **/home** directory that will be encrypted.

```
livecd-iso-to-disk --home-size-mb 1024    F10-i686-Live.iso  /dev/sbd1
```

If you do not want the data encrypted, add the **--unencrypted-home** option when creating the disk.

Combining both the overlay and the /home memory would use a command like the following. In all about 3 GB of disk space would be required.

```
livecd-iso-to-disk --overlay-size-mb 512 --home-size-mb 1024 Fedora-10-Live-
i686.iso  /dev/sbd
```

Creating Your Own Fedora Install Spins: Revisor and Pungi

You can create your own distribution discs with Pungi and Revisor. Pungi is the lower level spin generation tool developed by Red Hat. Revisor is a front end developed for Fedora by the Fedora Unity project which provides updated spins of the current Fedora release.

Revisor

The Fedora Unity Project provides re-spins of the current Fedora release, providing a CD/DVD Live image that has all the latest updates for a release. This way you can run a Live CD/DVD using the latest updated software for a given release. Should you decide to install Fedora, a Fedora Unity re-spin can avoid having to perform extensive downloads and updates. They are already incorporated into the re-spin. The Live-CD/DVD images are generated using Pungi and the livecd-tools. See **http://fedoraunity.org** for more information.

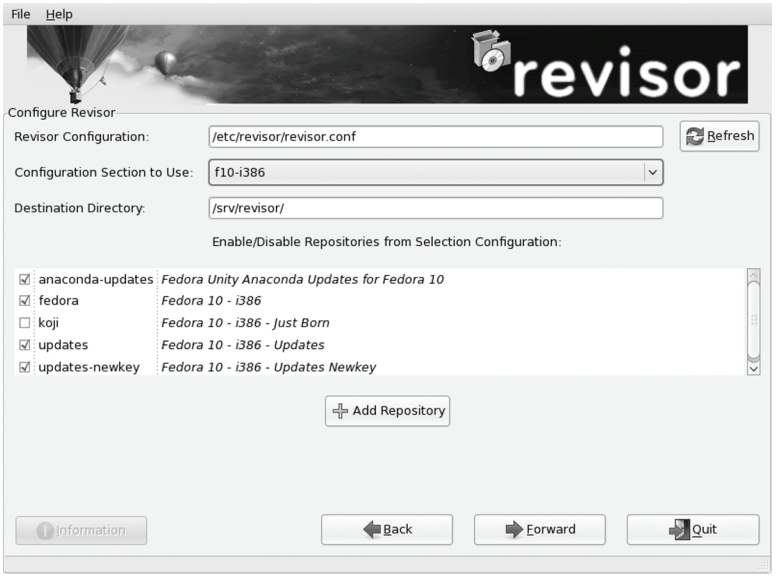


Figure B-5: Revisor configuration

Revisor is a GUI front end for Pungi and livecd-tools. You can use Revisor to create your own Live-CD/DVD disc image. Revisor will use Pungi to generate the Fedora spin, and then use livecd-tools to create the Live-CD/DVD image. Revisor is still very much a work in progress.

Use Add/Remove Software (PackageKit) to install Revisor (revisor packages). There are several packages: **revisor**, **revisor-gui**, **revisor-cli**, **revisor-unity**, and **revisor-comps**. Additional capability is provided by the **revisor-cobbler**, **revisor-resuseinstaller**, and **revisor-isolinux** packages**.** See the **http://revisor.fedoraunity.org/** for more details. You can also install revisor with a **yum** command. All dependent packages will be selected and installed for you.

```
yum install revisor
```

Revisor configuration files are located in the **/etc/revisor** directory. The **revisor.conf** file is the default configuration file for revisor. Repository information is kept in conf files located in the **/etc/revisor/conf.d** directory. Here you will find configuration file for different architectures like i386, x86_64, and rawhide (development), like **revisor-f10-i386.conf**. There is no configuration file that includes the RPM Fusion repository. You can add them easily on the Configuration page when you run Revisor.

To start Revisor, select the Revisor entry in the Applications | System Tool menu. Revisor starts up with a Welcome screen. Clicking the Get Started button then displays a dialog where you select the install media type and the live media type. Install media can be either CD or DVD. The live media can be either an optical disk image (USB is not yet supported).

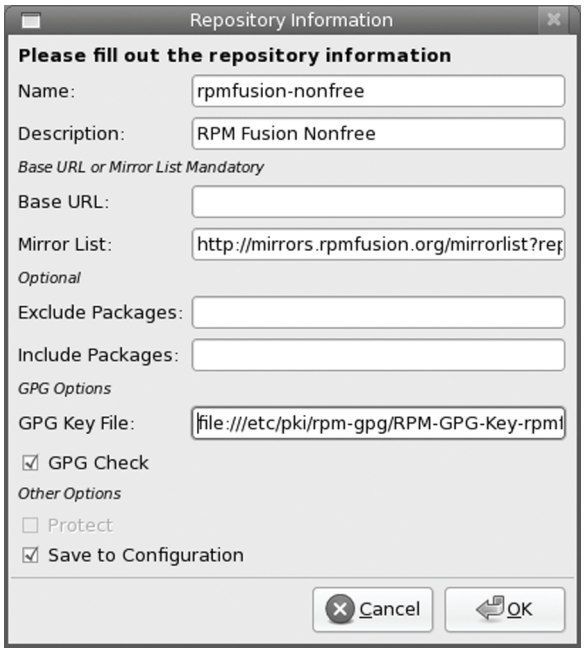


Figure B-6: Revisor added repository

The Revisor configuration dialog lets you choose the release for your spin. The i386 version will be selected by default (see Figure B-5). You can choose others, like the x86_64 or rawhide development versions. The default configuration file will be listed. You can specify a different one. Selected repositories will be displayed with checkboxes. You can uncheck ones you may not need.

You can also specify a different configuration file if you wish, or destination directory (default is **/srv/revisor**). Click the Refresh button to have another configuration file loaded.

To add a repository like RPM Fusion, click on the Add Repository button (see Figure B-6). Enter a repository name and a description. The name can be any name. You will also need to know either the Base URL or the Mirror List URL. Use one or the other, but not both. The Base URL for RPM Fusion Free repository for the i386 architecture is:

```
http://download1.rpmfusion.org/free/fedora/releases/10/Everything/i386/os/
```

The mirror list is at:

```
http://mirrors.rpmfusion.org/mirrorlist?repo=free-fedora-10&arch=i386
```

Be sure to specify the correct architecture. For an x86_64 architecture, substitute i386 with x86_64 in the previous examples.

Figure B-7: Revisor package selection

For the RPM Fusion Free repository GPG key you would use the following. Be sure you have already installed the RPM Fusion Free YUM configuration packages, which includes the GPG key.

```
file:///etc/pki/rpm-gpg/RPM-GPG-KEY-rpmfusion-free-fedora
```

To have the repository configuration saved to the Revisor conf file for that architecture, check the "Save to Configuration" box. Next time the repository will appear in the architecture's repository listing on the Revisor configuration window.

Add another repository for the RPM Fusion nonfree repository. Simply replace free with nonfree in the previous entries. You will have to know the repository URL for the RPM Fusion free and non free repositories, just as you would to add RPM Fusion access during installation.

A package selection screen is then displayed, with categories on the left pane, and packages to select on the right (see Figure B-7). Descriptions of packages are listed in the lower

pane. Click a category's checkbox to add its packages. The Options Packages button will then become active, if there are added packages to choose from. Click the Optional Packages button to open a window listing these packages (See Figure B-8).

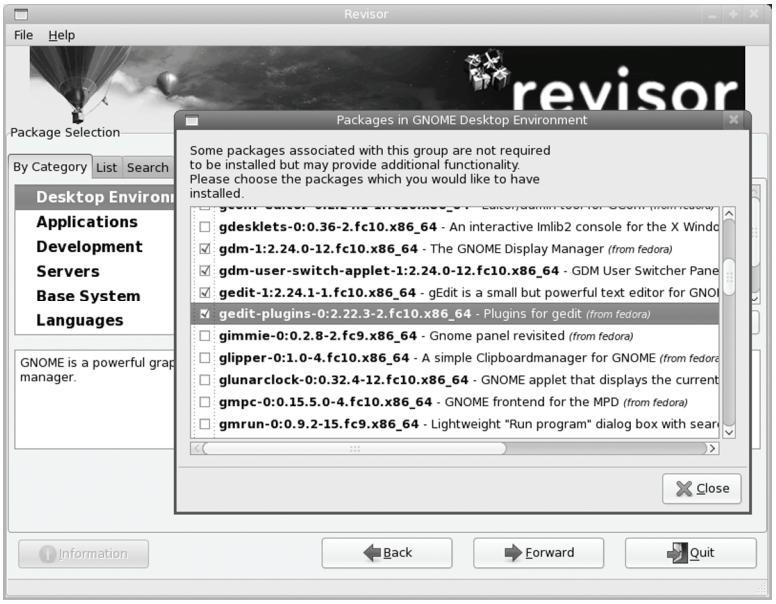


Figure B-8: Revisor optional packages

You can locate particular packages using the List or Find tabs. RPM Fusion packages will be identified in their description. The List and Search tabs let you locate individual packages. This is helpful if you already know the packages name.

RPM Fusion packages will be integrated into the package categories. Click the Optional Packages button for a particular category to see what RPM Fusion packages can be added in that category. You can also use Find and List to locate RPM Fusion Packages.

The size of the disk and number of selected packages are listed on the next screen.

Revisor then continues with the set of system-config-kickstart screen for configuration tasks such as a root user password, the time zone, automatic login, keyboard, authentication, firewall and SELinux, and a user account and password (see Figure B-9). As this spin is for a live-CD, there is no bootloader configuration.

Figure B-9: Revisor configuration set up tasks

The last screen will list the Revisor tasks being performed to create the disk image. Progress bars show the progress of the current task and the total progress. The first task will be the download of the updated packages. This may take some time.

The default location for the disk image is **/srv/revisor**. Here you will find the generated disk image.

Pungi

Pungi is the low level tool used to generate custom spins. Sample configuration files are located in the **/usr/share/pungi** directory. It will generate a basic collection of packages, downloading them from the Fedora repository. It uses the same syntax as kickstart. Pungi, like Revisor, can use the spin kickstart files located in the **/usr/share/spin-kickstarts** directory. The **spin-kicstarts** package installs several pre-configured kickstart configuration files for the Live CD spins currently provided by different groups. You can copy and edit these files to create your own configurations for a customized spin.

Table Listing

Figure Listing

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z