# *Excerpts of Chapters from*

# A Practical Guide to Ubuntu Linux®

FOURTH EDITION

**Mark G. Sobell**

**ISBN-13: 978-0-13-392731-3**






Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

BLANK

# 3

# Step-by-Step Installation



## In This Chapter



## Objectives

After reading this chapter you should be able to:

- Run a live session and use gnome-disks to view and change disk partitioning
- Install Ubuntu from a live session
- Install Ubuntu using the Server Image
- Modify system behavior using boot parameters
- Modify partitions during installation
- List the requirement and considerations for a dual-boot configuration

Chapter 2 covered planning the installation of Ubuntu: determining the requirements; planning the layout of the hard disk; obtaining the files you need for the installation, including how to download and burn or write Desktop and Server Images to installation media; and collecting information about the system. This chapter focuses on installing Ubuntu. Frequently the installation is quite simple, especially if you have done a good job of planning. Sometimes you might run into a problem or have a special circumstance; this chapter gives you tools to use in these cases. Read as much of this chapter as you need to; once you have installed Ubuntu, continue with Chapter 4, which covers getting started using the Ubuntu desktop and command line.

### Chapter 17 explains how to set up a virtual system

**tip** To install Ubuntu on a virtual system, build the virtual system and then follow the instructions in this chapter for installing the operating system. See page 691 for instructions on setting up a QEMU/KVM virtual machine and page 698 for setting up a VMware virtual machine.

### Upgrading an Ubuntu system

**tip** The easiest way to upgrade an Ubuntu system from one release to the next is to use the Software Updater window (page 123). Alternately, if you do not have an Internet connection you can boot from the installation medium and choose to upgrade from the Installation Type screen (page 64). Visit www.ubuntu.com/download/desktop/upgrade for more information.

### What to do if the installation does not work

**tip** On some hardware, the installation might pause for as long as ten minutes. Before experimenting with other fixes, try waiting for a while. If the installation hangs, try booting with one or more of the boot parameters described on page 75.



# BOOTING UBUNTU AND RUNNING A LIVE SESSION

Before Ubuntu can display the desktop of a live session or install itself on a hard disk, the Linux operating system must be read into memory (booted). This process can take a few minutes on older, slower systems and systems with minimal RAM (memory).

To boot the system, insert the installation medium holding a Desktop Image into the system and turn on or reset the system. Ubuntu starts booting. Refer to "BIOS setup" and "CMOS," both on page 32, if the system does not boot from the CD, DVD, or USB drive. Or refer to "Modifying Boot Parameters (Options)" on page 75 if Ubuntu does not boot or displays an error message.

## AUTOMATIC BOOT SEQUENCE

A few moments after it starts booting, Ubuntu displays the initial Boot screen, a mostly blank screen with keyboard layout and accessibility symbols at the bottom (Figure 3-1). After pausing for ten seconds (the default), the system boots. While it is booting, the system displays an Ubuntu logo and progress dots that turn on and off in sequence. The progress dots indicate the system is booting. If you do not interrupt the automatic boot sequence, Ubuntu boots to a live session (page 59).

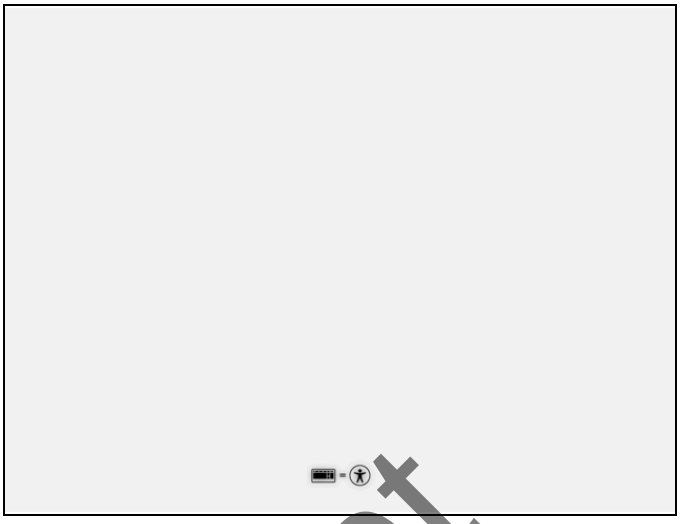

**Figure 3-1** The initial Boot screen

## DISPLAYING THE BOOT MENU

You must interrupt the automatic boot sequence to display the Boot menu. To interrupt the automatic boot sequence, press any key during the ten seconds that Ubuntu is displaying the initial Boot screen; Ubuntu displays the language overlay (Figure 3-2). Use the ARROW keys (the mouse will not work yet) to move the highlight to the language you want to use and press RETURN. Ubuntu displays the Desktop Boot menu (Figure 3-3, next page).

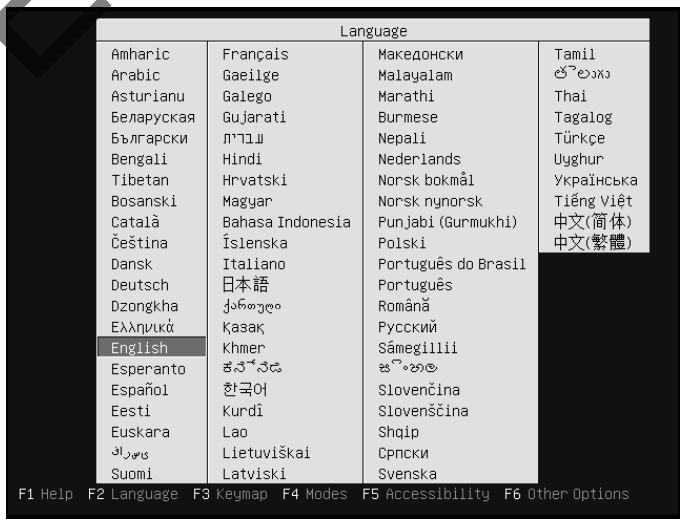


**Figure 3-2** The language overlay

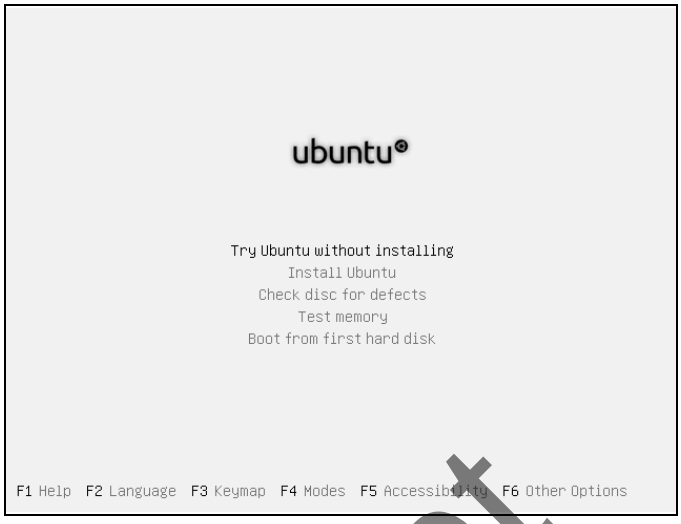


**Figure 3-3** The Desktop Boot menu



## TESTING THE INSTALLATION MEDIUM

The first time you use an installation medium, it is a good idea to check it for defects. Interrupt the automatic boot sequence and, from the Desktop Boot menu, use the ARROW keys to highlight **Check disc for defects** and press RETURN. Ubuntu checks the installation medium. If the installation medium is good, Ubuntu displays **Check finished: no errors found**; press any key to reboot the system. If the installation medium has errors, you must download and/or burn it again.

### Always test the installation medium

**caution** It is possible for data to become corrupted while fetching an installation image; it is also possible for a transient error to occur while writing an image to a recordable medium. Always follow the adjacent instructions to verify that the image on the installation medium does not contain any errors. Testing the installation medium takes a few minutes but can save you hours of aggravation if the installation fails due to a bad medium.

A DVD might fail the media test if the software that was used to burn the disk did not include padding. If a DVD fails the media test, try booting using the **nodma** parameter. See page 75 for information on adding parameters to the boot command line. If the DVD passes the media test when you boot the system using the **nodma** parameter, the DVD is good; reboot the system without this parameter before installing Ubuntu. If you install Linux after having booted using this parameter, the kernel will be set up to always use this parameter. As a consequence, the installation and operation of the system can be slow.

**optional**

## SEEING WHAT IS GOING ON

If you are curious and want to see what Ubuntu is doing as it boots from a Desktop Image, remove **quiet**, which controls kernel messages, and **splash**, which controls the Ubuntu logo screen, from the boot parameters. See Figure 3-16 on page 76; the list of parameters on the screen will be different from those in the figure. With the Desktop

Boot menu displayed, press F6 to display the boot command line and a drop-down list. Next press ESCAPE to close the drop-down list. Then press BACKSPACE or DEL to back up and erase **quiet** and **splash** from the boot command line. If you have not added anything to this line, you can remove the two hyphens at the end of the line. If you have added to this line, use the LEFT ARROW key to back up over—but not remove—whatever you added, the hyphens, and the SPACE on each side of them. Then remove **quiet** and **splash**. Press RETURN. Now, as Ubuntu boots, it displays information about what it is doing. Text scrolls on the screen, although sometimes too rapidly to read.

## Running a Live Session

As discussed in Chapter 2, a live session is a Linux session you run on a computer without installing Linux on the computer. When you reboot after a live session, the computer is untouched. If you are running Windows, after a live session Windows boots the way it did before the live session. If you choose to do so, you can install Ubuntu from a live session.

A live session gives you a chance to preview Ubuntu without installing it. Boot from a Desktop Image to begin a live session and work with Ubuntu as explained in Chapter 4. When you are finished, remove the installation medium and reboot the system. The system will boot as it did before you ran the live session.

Preserving files

Because a live session does not write to the hard disk (other than using a swap partition, if one is available), none of the work you save will be available once you reboot. You can use a USB flash drive, email, or another method to transfer files you want to preserve to another system. The unetbootin utility (page 52) can create persistent storage on a USB drive that you can write to from a live session.

Starting a live session

In most cases, you can boot Ubuntu to run a live session that displays the Unity desktop (Figure 3-4, next page) without pressing any keys after you boot from a Desktop Image. Ubuntu automatically logs in as the user named **ubuntu**. If you do not press any keys, Ubuntu displays the Welcome screen of the Install window (Figure 3-5, page 62). This window has two buttons: **Try Ubuntu** and **Install Ubuntu**; click **Try Ubuntu** to run a live session that displays the Unity desktop. If you pressed any keys as the system was booting, Unity displays the desktop immediately, without giving you the choice of trying or installing. Continue with Chapter 4.



# Basic Installation

This section describes how to install Ubuntu from a Desktop Image, both from a live session (preceding) and from the Boot menu (page 57).

### See what is on the hard disk (and back it up) before installing Linux

caution

Unless you are certain that the hard disk you are installing Ubuntu on has nothing on it (it is a new disk) or you are sure the disk holds no information of value, it is a good idea to examine the contents of the disk before you start the installation. You can use the gnome-disks utility from a live session to mount partitions on a hard disk. You can then examine the files in these partitions and see what is on the disk. See page 88 for more information on gnome-disks. Back up whatever is on the disk, even if it is in a partition the installation will not write to.


**Figure 3-4** The Unity desktop



**optional**

RAM Disks As part of the process of bringing up a live session or installing Ubuntu, ubiquity creates *RAM disks* (page 1268) that it uses in place of the hard disk used for a normal boot operation. The ubiquity installer copies tools required for the installation or to bring up a system from an installation image to the RAM disks. The use of RAM disks allows the installation process to run through the specification and design phases without writing to the hard disk and enables you to opt out of the installation at any point before ubiquity partitions the disk. If you opt out before this point, the system is left in its original state. The RAM disks also allow a live session to leave the hard disk untouched.

## INSTALLING FROM A LIVE SESSION

Bring up a live session as explained on page 59. Ubuntu will either display the Welcome screen with two icons, one of which depicts a DVD and has a button labeled **Install Ubuntu** below it, or a desktop with a hard disk icon labeled **Install Ubuntu**. Do nothing from the Welcome screen or double-click the hard disk icon to display the Welcome screen. With a Welcome screen displayed, continue reading at "The ubiquity Graphical Installer."

### The Ubiquity installer does not modify the live environment

**tip** The changes you make as you install Ubuntu from a live session do not affect the live session. For example, when you set up networking for the system you are installing, the network connection for the live environment does not change.

## INSTALLING FROM THE DESKTOP BOOT MENU

To install Ubuntu from a Desktop Boot menu, boot from the Desktop Image, interrupt the automatic boot sequence, and select a language as explained in "Displaying the Boot Menu" on page 57. Ubuntu displays the Desktop Boot menu (Figure 3-3, page 58). Use the ARROW keys to highlight **Install Ubuntu** and press RETURN. The system displays the Ubuntu logo and progress dots before displaying the Install window Welcome screen (Figure 3-5, next page). Refer to page 72 for information on using other Boot menu selections and page 73 for information on the function keys.

## THE ubiquity GRAPHICAL INSTALLER

Written mostly in Python, ubiquity is the graphical installer that runs by default when you install Ubuntu from a Desktop Image. Alternately, you can install Ubuntu using the textual installer (debian-installer; page 78).

The ubiquity installer identifies the hardware, loads drivers, probes for the devices it will use during installation, builds the filesystems, starts the X server, and installs the Ubuntu operating system.

Which screens ubiquity displays depends on which paths you pick through the installation process and which parameters you specify on the boot command line. Unless you tell it not to, ubiquity probes the video card and monitor, and starts a native X server.

Virtual consoles

While you are installing Ubuntu, ubiquity opens virtual consoles 1–7, with 7 reserved for the interactive installation. You can display a virtual console by pressing CONTROL-ALT-F*x*, where *x* is the virtual console number and F*x* is the function key that corresponds to the virtual console number.

At any time during the installation, you can switch to virtual console 2–6 (by pressing CONTROL-ALT-F2 through CONTROL-ALT-F6) and give shell commands to display information about processes and files. Do not give commands that change any part of the installation process. To switch back to the graphical installation screen, press CONTROL-ALT-F7. See page 127 for more information.

### USING THE MOUSE TO WORK WITH THE INSTALL WINDOW SCREENS

You can use either the mouse or the keyboard to make selections from the Install window screens. To select a language from the Welcome screen using the mouse, left-click the language you want to use in the list box at the left. If the language you want does not appear on the displayed portion of the list, use the scroll wheel to display more languages, then click the language you want to use. Ubuntu highlights the language

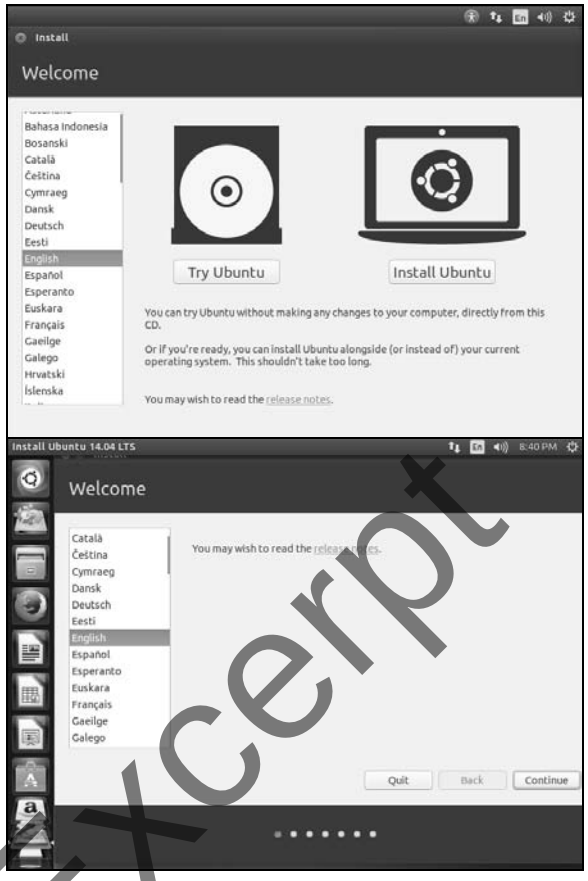


**Figure 3-5** The two Welcome screens



you click. Once you select a language, you are finished working with the Welcome screen. Click the button labeled **Continue** to display the next screen.

## USING THE KEYBOARD TO WORK WITH THE INSTALL WINDOW SCREENS

To use the keyboard to make selections, first use the TAB key to highlight the object you want to work with. On the Welcome screen, the objects are the selected item in the list box and the buttons labeled **Quit**, **Back**, and **Continue**.

List box With a language in the list box highlighted, use the UP ARROW and DOWN ARROW keys to highlight the language you want to use. The list scrolls automatically when you highlight the next, undisplayed entry in the list.

Button Once you select a language, you are finished working with the Welcome screen. Use the TAB key to highlight the button labeled **Continue**. The button has an orange border when it is highlighted. Press RETURN to display the next screen.

Drop-down list To make a selection from a drop-down list, such as the one in the box labeled **Mount point** shown in Figure 3-11 on page 69, use the TAB key to highlight the box and then use the ARROW keys to move the highlight from one item to the next. With the selection you want to choose highlighted, press RETURN.

## STARTING THE INSTALLATION

This book describes using the mouse to make selections from the graphical interface that ubiquity presents; you can use the keyboard if you prefer.

### WELCOME

The Welcome screen of the Install window (Figure 3-5) contains a list of languages to choose from. The language you choose will be the one ubiquity uses as you install the system and the default language for the installed system; you can change this default once the system is installed (see **Language Support** on the System Settings window, page 114). Select a language and click **Install Ubuntu** or **Continue**, depending on which button is displayed.

### PREPARING TO INSTALL UBUNTU

The Preparing to Install Ubuntu screen of the Install window verifies that the hard disk has enough space for you to use ubiquity to install the Desktop edition of Ubuntu (about 6.5 gigabytes) and that the system is connected to the Internet. If there is not enough space on the disk, the Continue button will be grayed out and you will not be able to continue. Lack of an Internet connection is not fatal.

The two check boxes on this screen are labeled

- **Download updates while installing**—A tick in this check box ensures that the installed system is up to date but will work only if the system is connected to the Internet. Without a tick in this check box the installation will proceed more quickly because ubiquity will not download files during the installation, but you will have to update the software on the newly installed system.
- **Install this third-party software**—Installs proprietary software that plays Flash, MP3, and other media, and works with certain graphics and Wi-Fi hardware.

Put ticks in the check boxes according to your preferences and click **Continue**.

### INSTALLATION TYPE

The Installation Type screen controls how ubiquity sets up the hard disk. It displays several radio buttons and check boxes. Exactly which ones it displays depends on the type of installation you are performing. It always includes a radio button labeled **Something else** at the bottom of the screen. Depressing this radio button causes ubiquity to display the Advanced Partitioning screen (page 67) next. This screen allows you to custom partition the disk. Following is a list of the radio buttons and check boxes that might appear on this screen. You can select only one radio button,

and certain radio buttons gray out the check boxes. See page 41 for a discussion of partitioning a hard disk.

- **Upgrade Ubuntu *XXX* to Ubuntu *XXX*—**Appears when ubiquity detects an earlier version of Ubuntu on the disk it is installing to. Selecting this option upgrades Ubuntu to the release that is on the installation medium.
- **Erase disk and install Ubuntu—**Appears when ubiquity did not detect an operating system on the disk it is installing to. Selecting this option deletes all information from the hard disk.
- **Erase Ubuntu *XXX* and reinstall—**Appears when ubiquity detects the same release of Ubuntu as it is installing on the disk it is installing to. Selecting this option deletes all information from the hard disk. This selection differs from **Reinstall Ubuntu** because it does not save any personal data.
  - **Encrypt the new Ubuntu installation for security—**Appears only when you have selected one of the preceding Erase buttons. This option encrypts the entire Ubuntu partition. The next step asks you to enter a passphrase (page 627; ubiquity calls it a *security key*) that you will have to enter each time you boot the system. If you lose/forget the passphrase, you will not be able to boot the system nor retrieve the data on the encrypted partition. This option forces you to use LVM (next).
  - **Use LVM with the new Ubuntu installation—**Appears only when you have selected one of the preceding Erase buttons. See page 46 for a discussion of LVM (Logical Volume Manager) including PVs, VGs, and LVs.
- **Something else—**Displays the Advanced Partitioning screen (page 67) that allows you to partition the disk or disks as you like, including resizing partitions.
- **Reinstall Ubuntu *XXX*—**Appears when ubiquity detects release ***XXX*** (the same release as you are installing) of Ubuntu on the disk it is installing to. This option keeps all personal data and attempts to reinstall all software that is already installed; it clears systemwide settings.
- **Install Ubuntu *XXX* along side *YYY*—**Appears when ubiquity detects operating system ***YYY*** on the disk to which it is installing version ***XXX*** of Ubuntu; sets up a dual-boot system (page 91). When you make this selection and the disk has enough free space (page 38) to install Ubuntu, ubiquity creates a partition and installs Ubuntu in that partition. If there is not enough free space to install Ubuntu, ubiquity shrinks an existing partition to make room for the new Ubuntu partition. Before shrinking the partition, ubiquity displays a screen that has a divider that you can use to adjust the amount of disk space allocated to each system. On this screen you can click **advanced partitioning tool** to display the Advanced Partitioning screen (page 67).

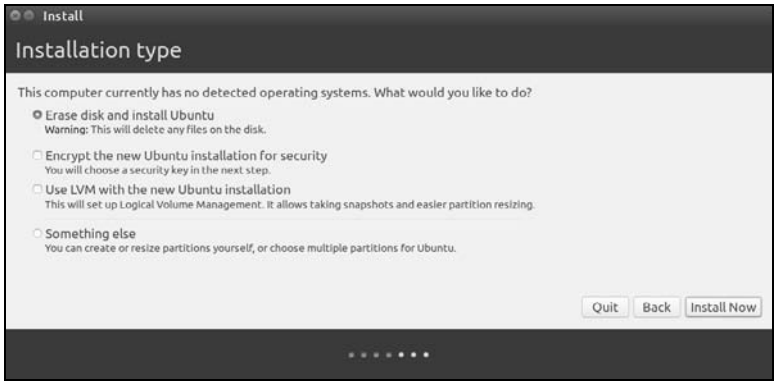


**Figure 3-6** The Install window, Installation Type screen 1

LPI Guided partitioning

With a single, clean hard disk—a hard disk with nothing installed on it, as it comes from the factory (i.e., no partition table)—ubiquity displays an Installation Type screen similar to the one shown in Figure 3-6. In this case, the simplest way to partition the disk, called *guided partitioning*, is to allow the ubiquity partitioner to do it for you (any selection except **Something else**). By default, the radio button labeled **Erase disk and install Ubuntu** is selected. If the system has two or more hard disks, the next step asks you to select the disk on which you want to install Ubuntu. Click **Install Now** (appears when you select **Erase disk and install Ubuntu**) or **Continue** (appears when you select **Something else**). When you choose to install Ubuntu, ubiquity creates two partitions on the hard disk: a small swap partition (page 42) and a root partition (/, page 42) that occupies the rest of the disk. See "The ubiquity Advanced Partitioning Screen" on page 67 if you select **Something else**.



### WHERE ARE YOU?

The Where Are You? screen allows you to specify the name of a city in the time zone where the computer is located. If the system is online, ubiquity will guess where you are and display that information. If needed, you can use the map or the text box to specify the time zone. When you click a location on the map, ubiquity moves the pin on the map to the location you clicked and displays in the text box the name of a city in the same time zone.

To use the text box to specify a city, erase the name in the text box and start typing the name of the large city you want to use to set the time zone; ubiquity displays a list of cities that match what you have typed so far. Click the city you want from the list and ubiquity fills in the text box and moves the pin on the map. Click **Continue**.

### KEYBOARD LAYOUT

The Keyboard Layout screen has two list boxes that allow you to specify the type of keyboard to be used by the installed system. The list box on the left allows you to specify a language and, in some cases, a location where the language is spoken. The list box on the right allows you to specify a keyboard type if more than one is available

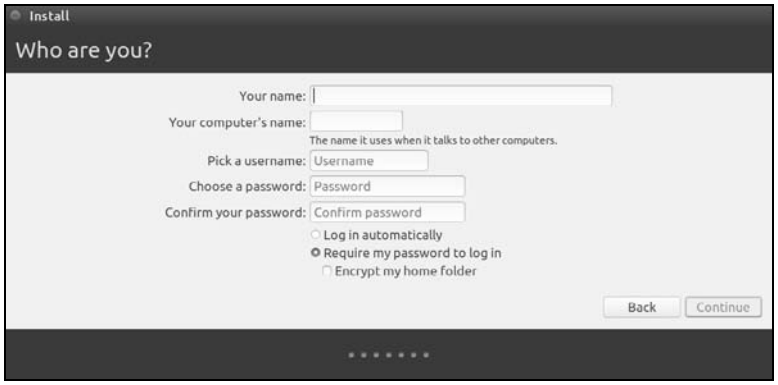


**Figure 3-7** The Install window, Who are You? screen 1

for the language you selected. See "F3 Keymap" on page 73 to change the layout of the keyboard ubiquity uses during installation.

Anytime the Keyboard Layout screen is displayed, you can highlight the text box at the bottom of the screen and type some letters to see whether the selected keyboard and language are correct for the keyboard you are using. Select a language and keyboard type from these list boxes and click **Continue**.

### WHO ARE YOU?

The Who Are You? screen (Figure 3-7) sets up the first Ubuntu user. This user can use sudo (page 98) to administer the system, including setting up additional users (page 564). Enter the full name of the user in the text box labeled **Your name**. As you type, ubiquity enters the lowercase version of the first name from the name you are entering in the boxes labeled **Your computer's name** (as a prefix) and **Pick a username**. Press TAB to move the cursor to either of these text boxes. If you want to use different values, press BACKSPACE (page 129) to erase the value and enter a new one. Press TAB.

For use on a local network and to connect to the Internet with a Web browser or other client, you can use a simple computer name such as **fox8**. If you are setting up a server system, see "FQDN" on page 894 for information on names that are valid on the Internet.

Enter the same password in the boxes labeled **Choose a password** and **Confirm your password**. The quality of the password is displayed to the right of the first box. Once you have entered the same password in both boxes, ubiquity displays a tick to the right of the lower box. Although ubiquity accepts any password, it is a good idea to choose a stronger (more secure) password if the system is connected to the Internet. For more information refer to "Password Security" on page 143.

The two radio buttons and the check box at the bottom of the window configure the login process for the user you are specifying. Select **Log in automatically** if you want Ubuntu to log you in automatically when the system boots—select this option only if you trust everyone who has physical access to the system. Select **Require my password to log in** to cause Ubuntu to require a password for you log in on the system.

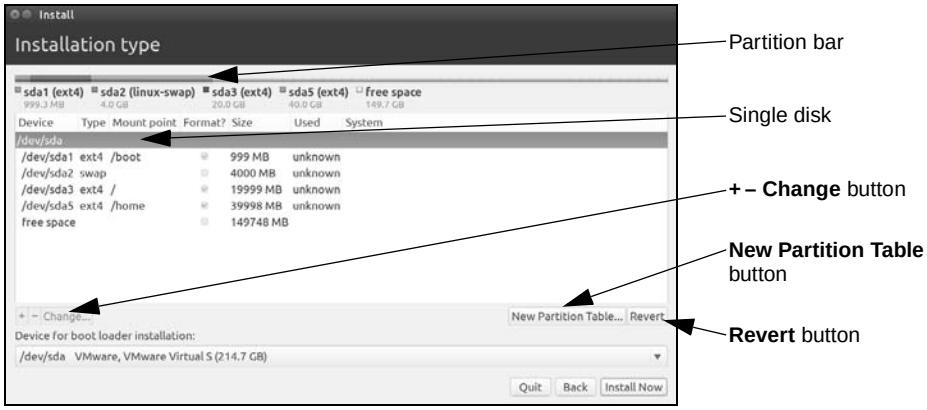


**Figure 3-8** The Advanced Partitioning screen showing a partitioned disk

Encrypt your home directory

Put a tick in the check box labeled **Encrypt my home folder** if you want ubiquity to set up an encrypted home folder. You cannot make this selection if you specify that you want to log in automatically. Click **Continue**.

The ubiquity installer displays the Install window as it installs Ubuntu. When ubiquity displays the Installation Complete dialog box, remove the installation medium and click **Restart Now**.



## LPI THE ubiquity ADVANCED PARTITIONING SCREEN

This section describes how to use the ubiquity Advanced Partitioning screen to partition a hard disk. See page 41 for a general discussion of partitioning.

You can display the Advanced Partitioning screen by clicking the radio button labeled **Something else** on the Installation Type screen (Figure 3-6, page 65) or by clicking **advanced partitioning tool** on the dual-boot divider screen (after selecting to install Ubuntu *along side of* another operating system).

The Advanced Partitioning screen in Figure 3-8 shows a disk with four partitions. The partition bar at the top of the window shows the relative sizes of the partitions, and the legend immediately below the bar shows the sizes of the partitions. The table below this information describes the partitions.

The Advanced Partitioning screen has the following headings:

- **Device**—The device name (page 481)
- **Type**—The filesystem type (Table 11-1 on page 497)
- **Mount point**—The name of the directory the filesystem will be mounted on (page 40)
- **Format?**—A tick in this check box indicates that the filesystem will be formatted; all data will be lost

BLANK

# 4

# Introduction to Ubuntu



## In This Chapter



## Objectives

After reading this chapter you should be able to:

- Log in on the Ubuntu desktop
- Understand **root** privileges
- Change the desktop background
- Use the Nautilus File Manager to work with files
- Correct mistakes on a command line
- Explain what you can do using a window titlebar
- Install and update software
- Display documentation using man, info, and yelp
- Open a terminal emulator and run programs from the command line
- Create and run a simple shell script

One way or another you are sitting in front of a computer that is running Ubuntu. After describing **root** privileges, this chapter takes you on a tour of the system to give you some ideas about what you can do with it. The tour does not go into depth about choices, options, menus, and so on; that is left for you to experiment with and to explore in greater detail throughout later chapters of this book.

The tour covers how to log in on the system, the Unity desktop, the Nautilus File Manager, the Settings window, getting help, and installing software. The final part of the chapter introduces some command-line utilities and describes how to work from the command line, concluding with a section on solving problems logging in and password security.

Be sure to read the warning about the dangers of misusing the powers of **root** (sudo) in the next section. While heeding that warning, feel free to experiment with the system: Give commands, create files, click objects, choose items from menus, follow the examples in this book, and have fun.

### root account

tip Most Linux systems include an account for a user named **root**. On a classic system a user can log in and work as **root** by providing the **root** password. As installed, Ubuntu has a **root** account but no password for the account: The **root** account is locked. The next section explains how you can use sudo and provide *your* password to run a command with **root** privileges. This book uses the phrase "working with **root** privileges" to distinguish this temporary escalation of privileges from the classic scenario wherein a user can work with **root** privileges for an entire session. See page 596 for more information on working with **root** privileges.



## CURBING YOUR POWER: root PRIVILEGES/sudo

When you enter your password to run a program (*not* when you log in on the system) or when you use sudo from the command line, you are working with **root** privileges and have extraordinary systemwide powers. A person working with **root** privileges is sometimes referred to as *Superuser* or *administrator*. While working with **root** privileges, you can read from or write to almost any file on the system, execute programs that ordinary users cannot, and more. On a multiuser system you might not be able to run certain programs. Nevertheless, someone—the *system administrator*—can, and that person maintains the system. When you are running Linux on your own computer, the first user you set up, usually when you install Ubuntu, is able to use sudo to run programs with **root** privileges.

### Who is allowed to run sudo?

security The first user you set up when you install Ubuntu is an administrator: This user can use sudo to execute any command. When you add user accounts, you can specify whether they are administrators. See Figure 4-16 on page 118 and page 596 for more information.

When this book says you have to enter your password, it assumes you have permission to administer the system. If not, you must get an administrator to perform the task.

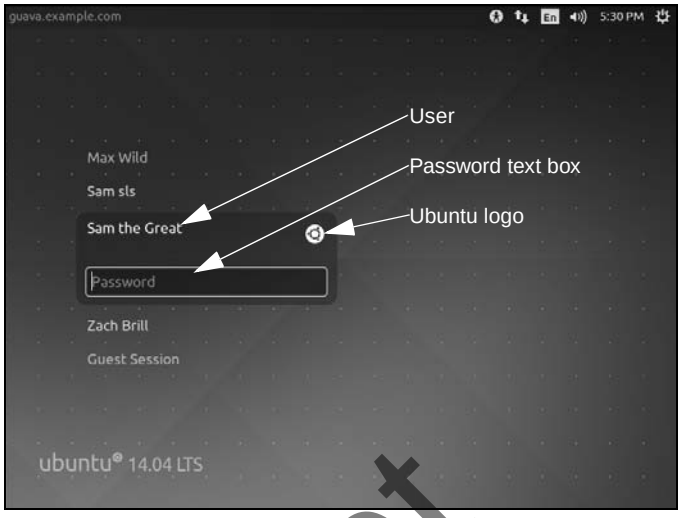


**Figure 4-1** The LightDM Login screen showing the Password text box

There are two primary ways to gain **root** privileges. First, when you start a program that requires **root** privileges, a dialog box pops up asking you to **Enter your password to perform administrative tasks**. After you enter your password, the program runs with **root** privileges. Second, if you use the sudo utility (page 602) from the command line (such as from a terminal emulator; page 126) and provide your password, the command you enter runs with **root** privileges. In both cases you cease working with **root** privileges when the command finishes or when you exit from the program you started with **root** privileges. For more information refer to "Running Commands with **root** Privileges" on page 596.



# LOGGING IN ON THE SYSTEM

### Do not experiment while you are working with root privileges

**caution** Feel free to experiment when you are *not* working with **root** privileges. When you *are* working with **root** privileges, do only what you have to do and make sure you know exactly what you are doing. After you have completed the task at hand, revert to working as yourself. When working with **root** privileges, you can damage the system to such an extent that you will need to reinstall Linux to get it working again.

When you boot a default Ubuntu system, LightDM (Light display manager; wiki.ubuntu.com/LightDM) displays a Login screen on the system console. In the middle of the screen is a list of names of users who can log in on the system. At the upper-right corner are icons that allow you to suspend, restart, or shut down the system, work with the network connection, adjust the volume, display a calendar, and display the universal access menu that allows you to change accessibility preferences (e.g., make the text larger or display a high-contrast desktop).

To log in, click your name; LightDM displays the Password text box (Figure 4-1). Enter your password and press RETURN. If LightDM displays an error message, try


**Figure 4-2** The Unity desktop

clicking your name and entering your password again. Make sure the CAPS LOCK key is not on (LightDM displays a message if it is) because the routine that verifies your entry is case sensitive. See page 142 if you need help with logging in and page 118 if you want to change your password. The system takes a moment to set things up and then displays the Unity desktop.



## THE UNITY DESKTOP

This section briefly describes how to use the Unity desktop that Ubuntu installs by default (Figure 4-2). Unity is a graphical shell for the GNOME desktop environment, so many Unity features compare with GNOME features. See "Installing the GNOME Flashback Desktop" on page 103 if you want to run the GNOME 2 (Classic) desktop or "Installing Other Desktop Environments" on page 124 if you want to install the GNOME 3 or another desktop.

When you log in on a system running Unity, the workspace is empty. On the Top panel (the bar at the top of the workspace) are objects: networking, language, speaker, time, and gear icons. You can click one of these objects to work with it. The Top panel remains constant as the information displayed below it changes.

### You can run the Classic desktop under Ubuntu

**tip** If you like, you can install and run the GNOME Classic desktop (Ubuntu calls it the GNOME Flashback desktop; it is GNOME 2). See "Installing the GNOME Flashback Desktop" on page 103.

This book uses the Linux textual interface (CLI) for most tasks, including setting up servers and programming in bash, SQL, and Python. Where this book describes a GUI utility, it explains how to open the utility/window by pressing ALT-F2 to display the Run a Command window and typing the name of the utility. Alternately, you can press the

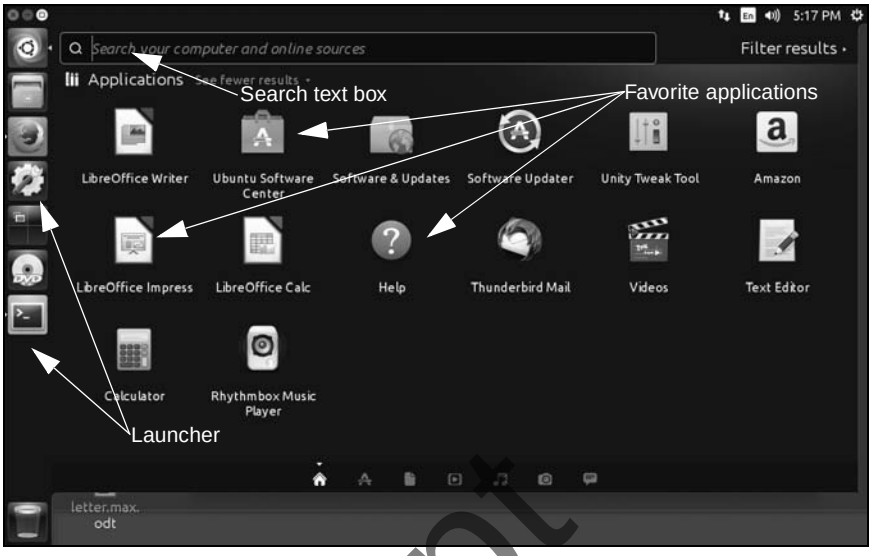


**Figure 4-3** The Dash

SUPER (Windows) key to display the Dash and enter the name of the utility there. For more information refer to "The Dash and the Run a Command Window" on page 106.

## THE DASH: RUNNING AN APPLICATION

Click the Ubuntu icon at the top of the Launcher on the left of a workspace (or press the SUPER [Windows] key) to display the Unity Dash[1] (simply called the Dash; Figure 4-3). The Dash holds the Search text box (near the top) with the Favorites list (icons of favorite applications) below it.

Running an application from the Favorites list

There are several ways to start an application. If the Dash is not visible, press the SUPER (Windows) key to display it. If the icon of the application you want to run is in the Favorites list, click that icon; Unity closes the Dash and opens the application whose icon you clicked.

Running an application from the Search text box

From the Dash you can also type the name of the (graphical) application (e.g., gnome-terminal or gnome-disks) or the name of the window the application displays (e.g., Terminal or Disks) in the Search text box and press RETURN to open that application in the active workspace.

### By default, the Dash searches the Internet

security

By default, when you enter text to search for in the Search text box in the Dash, the Dash searches the Internet in addition to the local system. You can turn off the online searching from the Search tab of the Security & Privacy window that you can display from the System Settings window (page 113).

1. Do not confuse the Unity Dash graphical screen with the dash shell (page 335).

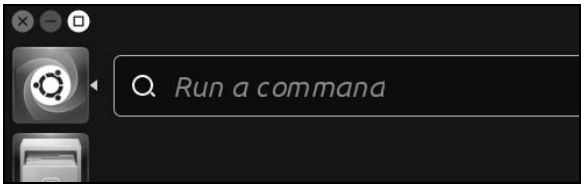


**Figure 4-4** A corner of the Run a Command window

## THE RUN A COMMAND WINDOW

Alternately, you can press ALT-F2 to display the Run a Command window (Figure 4-4), type the name of the application or utility you want to run, and press RETURN to run an application. You must run a textual application from a terminal emulator such as gnome-terminal (which you can start from a Run a Command window). See "The Dash and the Run a Command Window" on page 106 for more information.

**optional**

unity-tweak-tool

The unity-tweak-tool utility (**unity-tweak-tool** package) is a settings manager for the Unity desktop. You can use it to change many aspects of how the desktop looks and works. To install unity-tweak-tool, run the Terminal utility from the Dash as explained on the previous page; Terminal displays a terminal emulator window. With the cursor over the terminal emulator, type the following command; terminate the command by pressing RETURN.

```
$ sudo apt-get -y install unity-tweak-tool
[sudo] password for sam:
...
$
```

The sudo utility prompts for your password. Once you enter your password and press RETURN, apt-get displays information as it installs the **unity-tweak-tool** software package. When the shell displays its prompt (**$**), the package is installed. Once unity-tweak-tool is installed, you can run it as explained in the previous section. For more information on using apt-get to install packages see the JumpStart on page 512.



## THE APPLICATION SWITCHER: CHANGING THE INPUT FOCUS

The window with the *input focus* is the one that receives keyboard characters and commands you type. In addition to using the mouse, you can change which window has the input focus by using the keyboard; this process is called *window cycling*.

When you press ALT-TAB while the desktop is displayed, Unity displays in the center of the workspace the Application Switcher, a box that holds icons representing the programs running in the windows on the desktop. It also shifts the input focus to the window that was active just before the currently active window, making it easy to switch back and forth between two windows. When you hold ALT and press TAB multiple times, the focus moves from window to window. Holding ALT and SHIFT and repeatedly pressing TAB cycles windows in the other direction.

When you have several windows, each running the same utility (e.g., Terminal), pressing ALT-TAB will initially display only one of these windows. To select one window from a group of several that are running the same utility, hold ALT, press TAB repeatedly until the icon of the utility is highlighted, and then release TAB while still holding ALT; Unity expands the single icon to multiple icons, each displaying a window of an instance of the utility. Continue to hold ALT and press TAB repeatedly to cycle through these icons.

## INSTALLING THE GNOME FLASHBACK DESKTOP

If you prefer to work with a GNOME 2 desktop, also called the GNOME Classic or GNOME Flashback desktop, you can install it and select either Unity or GNOME Flashback each time you log in.

The following instructions explain how to install and run the GNOME Classic desktop on an installed Ubuntu system. You cannot perform this task on a live session.

1. With Ubuntu installed and running the Unity desktop, click the Ubuntu logo at the top of the Launcher at the upper left of the screen; Unity displays the Search text box. Type **terminal** and press RETURN; Unity displays a terminal emulator window.
2. With the cursor over the terminal emulator, type the following command; terminate the command by pressing RETURN.

```
$ sudo apt-get update
[sudo] password for sam:
...
$ sudo apt-get -y install gnome-session-flashback
...
$
```

   The sudo utility prompts for *your* password. Once you enter your password and press RETURN, apt-get displays a lot of information as it updates the package database and then as it installs the **gnome-session-flashback** software package. When the shell displays its prompt (**$**) after giving the second command, the package is installed. For more information on using apt-get to install packages see page 512.
3. Log out by clicking the gear at the upper-right corner of the screen, clicking **Log Out** from the menu Unity displays, and then clicking the button labeled **Log Out** from the small window Unity displays. After a few moments Unity displays a Login screen (Figure 4-1, page 99).
4. Click your name on the Login screen; Unity displays the Password text box (Figure 4-1, page 99). To the right of your name is an Ubuntu logo. Click this logo to display the Select Desktop Environment menu (Figure 4-5, next page). This menu gives you the choice of using the Compiz (compiz.org) or Metacity window manager with GNOME Flashback (the classic GNOME desktop). Click one of the **GNOME Flashback** selections (Metacity allows you to have multiple workspaces); Unity restores the Password text box.

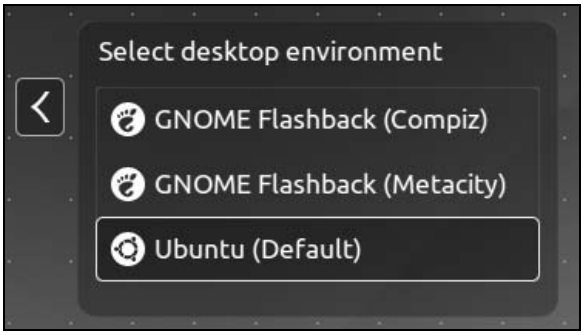


**Figure 4-5** The Select Desktop Environment menu on the Login screen

5. Type your password in the Password text box and press RETURN. After a few moments the system displays a GNOME Flashback desktop (Figure 4-6).

Once you have selected GNOME Flashback when you log in, it is your default desktop; you do not have to select it next time you log in.

### THE GNOME FLASHBACK DESKTOP

This section briefly describes the desktop that is installed in the previous section.

Panels and objects

When you log in, the GNOME Flashback desktop displays a workspace that includes the Top and Bottom panels (bars) that help you get your work done easily and efficiently (Figure 4-6). Each of the panels can hold several icons and words called *objects*.

The Applications menu

You can run an application from the Applications menu that appears at the left end of the Top panel (Figure 4-6). Click **Applications** to display the Applications menu. When you move the mouse pointer over one of the selections in this menu and leave it there for a moment (this action is called *hovering*), the menu displays a submenu. Whereas it used to be a top-level menu, **Administration** is now a submenu of **System Tools**. Move the cursor to and click one of the items in a submenu to run it.

The Places menu

Click an entry in the Places menu to select it. The Places menu on the GNOME Flashback desktop is the same as Places in the Nautilus File Browser Sidebar; see page 109 for more information.

## WORKING WITH THE UNITY DESKTOP

This section discusses several ways you can work with the Unity desktop. It assumes that you have enabled workspaces as described in the following tip.

### Enabling workspaces

tip

As installed, workspaces are disabled in Unity. To enable workspaces, click the gear at the upper-right corner of the desktop and select **System Settings** from the menu. From the System Settings window, click **Appearance** (at the top left of the window), click the Behavior tab in the Appearance window, and put a tick in the check box labeled **Enable workspaces**. Close the Appearance window. Use the workspace switcher icon (on the Launcher) to display different workspaces.

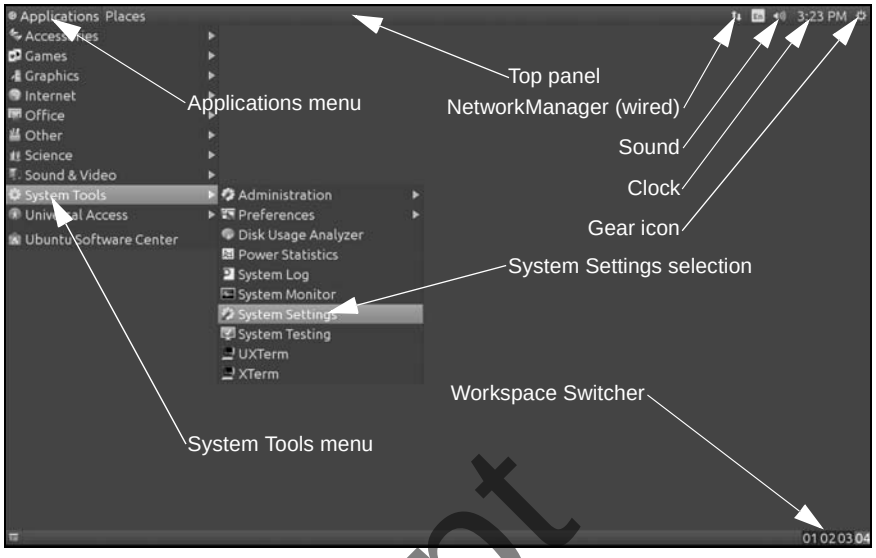


**Figure 4-6** The GNOME Flashback desktop showing the **System Tools⇨System Settings** menu



## TERMINOLOGY

In addition to this section, see the Glossary, which defines data entry *widgets* (page 1281) such as the *combo box* (page 1240), *drop-down list* (page 1245), *list box* (page 1257), and *text box* (page 1277).

Workspace — A *workspace* is a screen that holds windows of one or more applications. When workspaces are enabled (see the preceding tip), the Workspace Switcher icon on the Launcher enables you to display any workspace.

Active workspace — The *active workspace* is the workspace that displays the windows you can work with immediately. Only one workspace can be active at a time.

Desktop — The *desktop,* which is not displayed all at once, is the collection of all workspaces.

Panel — Panels are bars that can appear on the desktop and hold objects. The Top panel appears at the top of the screen; the Bottom panel, if it is present, appears at the bottom of the screen.

Launcher — The Launcher is the vertical stack of icons at the left of the workspace (Figure 4-3, page 101). Click one of the icons to run a program or open a window.

Object — An *object* is a word, icon, or menu that you can select by clicking.

### Click and right-click

tip — This book uses the term **click** when you need to click the *left* mouse button. It uses the term **right-click** when you need to click the *right* mouse button. See page 116 for instructions on adapting the mouse for left-handed use.

## THE DASH AND THE RUN A COMMAND WINDOW

You can start a graphical application (open an application window) by pressing the SUPER (Windows) key to display the Dash (Figure 4-3, page 101) or by pressing ALT-F2 to display the Run a Command window (Figure 4-4, page 102) and typing the name of the application. Try typing **firefox** and pressing RETURN to start Firefox. The new window opens on the active workspace.

### You must run a textual application (utility) from a terminal emulator

**tip** sA textual application or utility requires a textual environment such as a terminal emulator to run. The Dash and the Run a Command window do not provide a textual environment and so cannot run a textual application. However, you can use either of these windows to run gnome-terminal, a graphical application that displays the Terminal window and provides a textual environment. You can then run the textual application from the terminal emulator that gnome-terminal displays.

## CONTEXT MENUS

A *context menu* has choices that apply specifically to the window or object you click. The choices differ from window to window and from object to object. Some windows do not have context menus. Frequently a right-click displays a context menu.

### Right-click to display a context menu

**tip** Right-click an object to display its context menu. Each object displays its own context menu, although similar objects have similar context menus. Most context menus have either a Preferences or Properties selection.

## WINDOWS

Window In a workspace, a *window* is a region that runs, or is controlled by, a particular program.

Titlebar A *titlebar* (Figure 4-7) appears at the top of most windows and controls the window it is attached to. You can double-click the titlebar to maximize and restore a window. Clicking the close button (**X**) closes the window and usually terminates the program running in it. To reposition the window, click the titlebar and drag the window to the desired location.

Window Operations menu Right-click the titlebar to display the Window Operations menu. This menu allows you to minimize, maximize, resize, or close a window; move the window within the workspace; move a window to another workspace; keep the window on top of or below other windows; and cause the window to always be visible on the displayed workspace.

Menubar The *menubar* (Figure 4-7) holds icons that you can use to work with the window contents. As installed, Unity positions the menubar on the Top panel; see the following tip.

### The menubar can appear on the Top panel or on each window

**tip** As installed, Unity positions the menubar on the Top panel. You can choose to have Unity display a menubar at the top of each application window: Click the gear at the upper right of the desktop, select **System Settings**, click **Appearance**, click the **Behavior** tab, and, in the frame labeled Show the Menus for a Window, select the radio button labeled **In the window's title bar**. With this setup, Unity displays the name of the window (the classic titlebar) but, when you hover the mouse pointer over the titlebar, changes the display to the menubar.

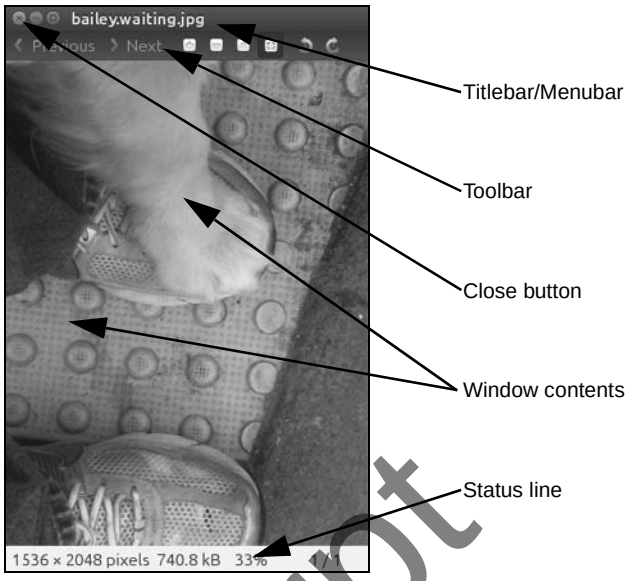


**Figure 4-7** A typical window

Toolbar

A *toolbar* (Figure 4-7) usually appears near the top of a window and contains icons, text, applets, menus, and more. Many kinds of toolbars exist. The titlebar is not a toolbar.

Resizing a window

To resize a window, position the mouse pointer over an edge of the window; the pointer turns into an arrow pointing to a line. When the pointer is an arrow pointing to a line, you can click and drag the side of a window. When you position the mouse pointer over a corner of the window, you can resize both the height and the width of the window simultaneously. Some windows are not resizeable and some windows do not allow you to resize them from all four sides.

Moving a window

To move a window, click and drag the titlebar. Dragging a window to the top of the workspace maximizes the window.



## CUTTING AND PASTING OBJECTS USING THE CLIPBOARD

There are two similar ways to cut/copy and paste objects and text both within and between windows. In the first method, you use the clipboard, technically called the *copy buffer*, to copy or move objects or text. To do so, you explicitly copy an object or text to the buffer and then paste it somewhere else. Applications that follow the user interface guidelines use CONTROL-X to cut, CONTROL-C to copy, and CONTROL-V to paste. Application context menus frequently provide these options.

You might not be familiar with the second method to copy and paste text—using the *selection* or *primary* buffer, which always contains the text you most recently selected (highlighted). You cannot use this method to copy objects. Clicking the middle mouse button (click the scroll wheel on a mouse that has one) pastes the contents of the selection buffer at the location of the mouse pointer. If you are using a two-button mouse, click both buttons at the same time to simulate clicking the middle button.

With both of these techniques, start by highlighting an object or text to select it. You can drag a box around multiple objects to select them or drag the mouse pointer over text to select it. Double-click to select a word or triple-click to select a line or a paragraph.

Next, to use the clipboard, explicitly copy (CONTROL-C) or cut (CONTROL-X) the objects or text. If you want to use the selection buffer, skip this step.

To paste the selected objects or text, position the mouse pointer where you want to put it and then either press CONTROL-V (clipboard method) or press the middle mouse button (selection buffer method).

### Use SHIFT-CONTROL-C and SHIFT-CONTROL-V within a terminal emulator

tip The CONTROL-C, CONTROL-X, and CONTROL-V characters do not work in a terminal emulator window because the shell running in the window intercepts them before the terminal emulator can receive them. However, you can use SHIFT-CONTROL-C and SHIFT-CONTROL-V in place of CONTROL-C and CONTROL-V, respectively. There is no keyboard shortcut for CONTROL-X. You can also use the selection buffer in this environment or use copy/paste from the **Edit** selection on the menubar or from the context (right-click) menu.

When using the clipboard, you can give as many commands as you like between the CONTROL-C or CONTROL-X and CONTROL-V, as long as you do not press CONTROL-C or CONTROL-X again. When using the selection buffer, you can give other commands after selecting text and before pasting it, as long as you do not select (highlight) other text.

## LOGGING OUT

Log out by clicking the gear at the upper-right corner of the workspace, clicking **Log Out** from the menu Unity displays, and then clicking the button labeled **Log Out** from the small window Unity displays. Select **Shut Down** or **Restart** from the gear menu to shut down or restart the system, respectively. See "Bringing the System Down" on page 441 for ways to shut down and restart the system from the command line.



# USING THE NAUTILUS FILE MANAGER

Nautilus (the Files program) is the simple, powerful Unity file manager. You can use it to create, open, view, move, and copy files and folders as well as to execute applications and scripts.

Terms: folder and directory

Nautilus displays the File Browser window, which displays the contents of a folder. The terms *folder* and *directory* are synonymous; "folder" is frequently used in graphical contexts, whereas "directory" might be used in textual or command-line contexts. This book uses these terms interchangeably.

Term: File Browser

This book sometimes uses the terms *File Browser window* and *File Browser* when referring to the Nautilus File Browser window.

Opening Nautilus

Give the command **nautilus** from a Run a Command window (ALT-F2) or a terminal emulator to open a Nautilus File Browser window that shows the files in your home directory (Figure 4-8). Alternately, you can type **nautilus** or **files** in the Dash or you can click the filing cabinet icon in the Launcher.

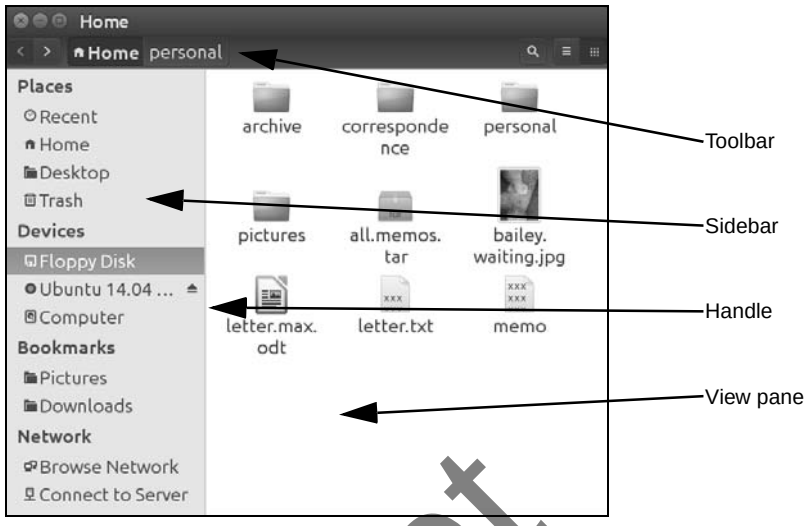


**Figure 4-8** A Nautilus File Browser window

## THE NAUTILUS FILE BROWSER WINDOW

The right side of a File Browser window shows the View pane that holds file objects; the left side shows the Sidebar that holds the Places, Devices, Bookmarks, and Network lists.

View pane The View pane displays file icons or a list of filenames. Select the view you prefer by clicking the dots or lines button near the right end of the toolbar.

Sidebar The Sidebar allows you to work with Nautilus in different ways. Close or display the Sidebar by pressing F9. To change the horizontal size of the Sidebar, drag the handle (Figure 4-8) on its right side. See "The Sidebar" (next) for more information.

Toolbar The left end of the toolbar allows you to display different directories in the View pane. Press CONTROL-L or select **Files menu: Go⇨Enter Location** to open a location bar in which you can enter a pathname.

## THE SIDEBAR

The Places section of the Sidebar holds objects that you click to change what Nautilus displays in the View pane. The Home and Desktop objects display your directories with corresponding names. The Computer object displays the local filesystem. The Browse Network object displays systems and files on the local network. The Connect to Server object opens a window that allows you to enter a URL to connect to an SSH, FTP, or other type of server.

Trash Selecting **Move to Trash** from an object's context (right-click) menu moves the object to the **Trash** directory. Because files in the trash take up space on the hard disk (just as all files do), it is a good idea to remove them periodically. To display the **Trash** directory, double-click the Trash icon in the Sidebar.

Emptying the trash Select **Empty Trash** from the Trash icon context (right-click) menu to permanently remove all files from the trash. (This selection is grayed out if there are no files in the

trash.) Alternately, you can right-click an object in the Trash File Browser window and select **Delete** to remove only that object (file), or you can select **Restore** to move the file back to its original location. You can drag and drop files to and from the trash just as you can with any folder.

Deleting files

You can delete files without first sending them to the trash. Add **Delete** to the context menu by selecting **Files menu: Edit⇨Preferences**, then selecting the Behavior tab, and putting a tick in the check box labeled **Include a Delete command that bypasses Trash**.

### Nautilus can open a terminal emulator

**tip** When you install the **nautilus-open-terminal** package (see page 512 for instructions) and log out and log back in, Nautilus presents an Open in Terminal selection in context menus where appropriate. For example, with this package installed, when you right-click a directory object and select **Open in Terminal**, Nautilus opens a terminal emulator with that directory as the working directory (page 151).

## OPENING FILES

By default, you double-click a file object in the View pane to open it; alternately, you can right-click the object and select **Open** from the drop-down list. When you open a file, Unity figures out the appropriate tool to use by determining the file's *MIME* (page 1259) type. Unity associates each filename extension with a MIME type and each MIME type with a program. Initially Unity uses the filename extension to try to determine a file's MIME type. If it does not recognize the filename extension, it examines the file's *magic number* (page 1258).

For example, when you open a file with a filename extension of **ps**, Unity calls the Evince document viewer, which displays the PostScript file in a readable format. When you open a text file, Unity uses gedit to display and allow you to edit the file. When you open a directory, Unity displays its contents in a File Browser window. When you open an executable file such as Firefox, Unity runs the executable file. When Unity uses the wrong tool to open a file, the tool generally issues an error message. See "Open With" on page 112 for information on how to use a tool other than the default tool to open a file.

From within a File Browser window, you can open a folder in a new tab. To do so, middle-click the folder or right-click the folder and select **Open in New Tab** from the drop-down list; Nautilus displays a new tab named for the folder you clicked. Click the new tab to display contents of the directory.

## SELECTING OBJECTS

Within a File Browser window, select an object by clicking it once; Unity highlights the object. Select additional objects by holding down the CONTROL key while you click each object. To select a group of adjacent objects, highlight the first object and then, while holding down the SHIFT key, click the last object; Unity highlights all objects between the two objects you clicked. Alternately, you can use the mouse pointer to drag a box around a group of objects.

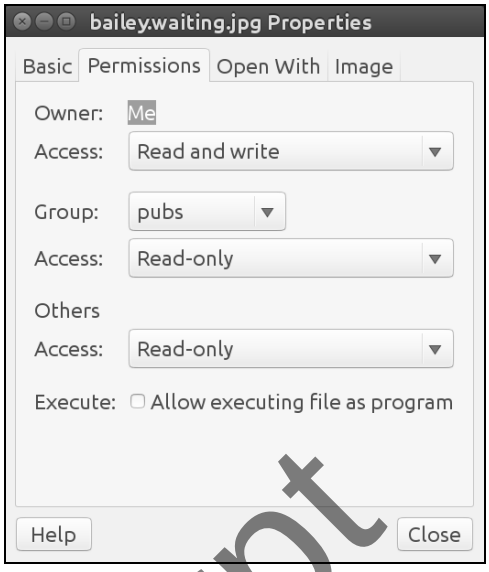


**Figure 4-9** The Object Properties window, Permissions tab

## THE OBJECT PROPERTIES WINDOW

The Object Properties window displays information about a file, such as its owner, permissions, size, location, MIME type, ways to work with it, and so on. This window is titled ***filename* Properties**, where ***filename*** is the name of the file you clicked to open the window. To display this window, right-click an object and select **Properties** from the drop-down list. The Properties window initially displays some basic information. Click the tabs at the top of the window to display additional information. Different sets of tabs appear for different types of files. You can modify the settings in this window only if you have permission to do so. This section describes the three tabs most commonly found in Object Properties windows.

Basic The Basic tab displays information about the file, including its *MIME* (page 1259) type, and enables you to select a custom icon for the file and change its name. To change the name of the file, replace the name in the text box labeled **Name**. If the filename is not listed in a text box, you do not have permission to change it. An easy way to change the icon is to open a File Browser window at **/usr/share/icons** (give the command **nautilus /usr/share/icons** in a Run a Command window [ALT-F2]). Work your way down through the directories until you find an icon you like (**gnome/24x24** has some interesting icons), and then drag and drop it on the icon to the left of **Name** in the Basic tab.

Permissions The Permissions tab (Figure 4-9) allows the owner of a file to change the file's permissions (page 199) and to change the group (see **/etc/group** on page 484) the file is associated with to any group the owner is associated with. Nautilus grays out items you are not allowed to change.

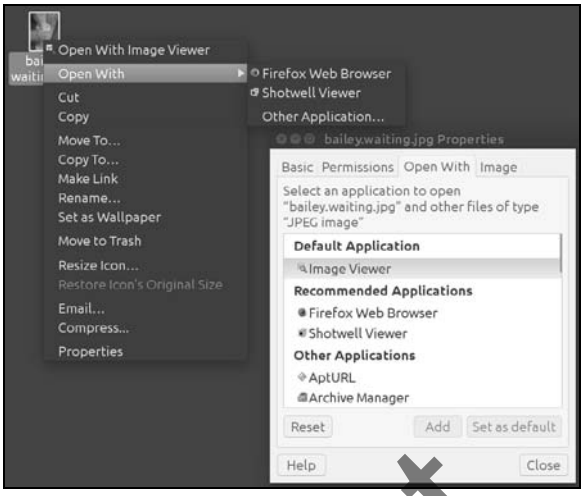


**Figure 4-10** The Object Properties window, Open With tab, and the context menu, Open With submenu for the same file



Using the drop-down lists, you can give the owner (called *user* elsewhere; see the tip about chmod on page 202), group, and others read and write or read-only permission for a file. Alternately, you can prohibit the group and others from accessing the file by specifying permissions as **None**. Put a tick in the check box labeled **Execute** to give all users permission to execute the file. This tab does not give you the fine-grained control over assigning permissions that chmod (page 201) does.

Permissions for a directory work as explained on page 205. Owner, group, and others can be allowed to list files in a directory, access (read and—with the proper permissions—execute) files, or create and delete files. Group and others permissions can also be set to **None**. Click **Change Permissions for Enclosed Files** to change the permissions of all files in a directory.

Open With

When you ask Nautilus to open a file that is not executable (by double-clicking its icon or right-clicking and selecting the first **Open with** selection), Nautilus determines which application or utility it will use to open the file. Nautilus uses several techniques to determine the *MIME* (page 1259) type of a file and selects the default application based on that determination.

The Open With tab (Figure 4-10) enables you to change which applications Nautilus can use to open the file and other files of the same MIME type (typically files with the same filename extension). Scroll the list of applications, highlight one, and click the **Add** button to add to the list of applications. Right-click an application and select **Forget association** to remove an application from the list. You cannot remove the default application.

When you add an application, Nautilus adds that application to the Open With list but does not change the default application it uses to open that type of file. Highlight

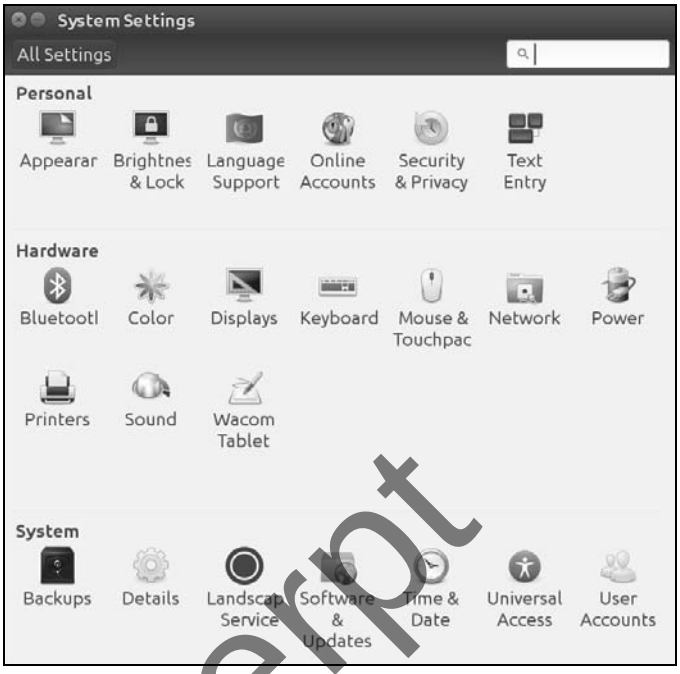


**Figure 4-11** The Systems Settings window

the application and click **Set as default** to cause that application to become the default application Nautilus uses to open this type of file.

When a file has more than one application in the Open With tab, the context menu displays an Open With submenu (Figure 4-10).



# THE SYSTEM SETTINGS WINDOW

The System Settings window (Figure 4-11) enables you to customize your account so it better meets your needs. Click the gear at the upper right of the desktop and then click **System Settings** to display this window. Alternately, you can open this window by giving the command **unity-control-center** from a Run a Command window (ALT-F2) or a terminal emulator or by clicking the gear icon in the Launcher. This section discusses the items in this window. Click an item to display its corresponding window.

### You might need to unlock a window before you can change settings

**tip** In the upper right of some windows there is an unlocked lock icon with the word **Unlock** next to it. See Figure 4-16 on page 118 for an example. When you click this icon, Unity displays an Authentication Required dialog box that requests your password. Enter your password and click **Authenticate** to unlock the window.

BLANK

# 7

# The Linux Utilities



## In This Chapter



## Objectives

After reading this chapter you should be able to:

- Use basic utilities to list files and display text files
- Copy, move, and remove files
- Display the beginning or end of a file
- Search, sort, print, categorize, and compare text files
- Compress, decompress, and archive files
- Count the number of letters, words, and lines in a file
- Locate utilities on the system
- Change the modification time of a file
- Display information about users and the system
- Record a session in a file
- Edit text files

When Linus Torvalds introduced Linux and for a long time thereafter, Linux did not have a GUI (graphical user interface): It ran on character-based terminals only, using a CLI (command-line interface), also referred to as a textual interface. All the tools ran from a command line. Today the Linux GUI is important, but many people—especially system administrators—run many command-line utilities. Command-line utilities are often faster, more powerful, or more complete than their GUI counterparts. Sometimes there is no GUI counterpart to a textual utility; some people just prefer the hands-on feeling of the command line. This chapter describes a number of frequently used utilities and concludes with tutorials on the vim and nano text editors. See "Working from the Command Line" on page 125 for an introduction to the command line.

When you work with a CLI, you are working with a shell (Chapters 5, 9, and 28). When you are working on the command line it is important to quote special characters as explained on page 150.

**More utilities are covered throughout the book**

tip This chapter introduces a few important utilities that are good to know as you start using Linux. More utilities are covered throughout the book. See the inside of the front and back covers for a complete list.



# BASIC UTILITIES

One of the advantages of Linux is that it comes with thousands of utilities that perform myriad functions. You will use utilities whenever you work with Linux, whether you use them directly by name from the command line or indirectly from a graphical menu or icon. The following sections discuss some of the most basic and important utilities that are available from a CLI. Some of the more important utilities are also available from a GUI; others are available only from a GUI.

**Run these utilities from a command line**

tip This chapter describes command-line, or textual, utilities. You can experiment with these utilities from a terminal, a terminal emulator within a GUI (page 126), or a virtual console (page 127).

## LPI cat: JOINS AND DISPLAYS FILES

The cat utility joins and copies files to standard output. The name of the cat utility is derived from *catenate*, which means to join together, one after the other. You can also use cat to display virtual system files in the **/proc** directory hierarchy as explained on page 490.

### ARGUMENTS

The arguments are the pathnames of one or more files that cat processes. You can use cat to display the contents of one or more text files on the screen. More precisely, cat copies the files to standard output, which by default is attached to the screen.

```
$ cat practice
This is a small file that I created
using:  a text editor!

End.
```

If you do not specify an argument or if you specify a hyphen (**–**) in place of a filename, cat reads from standard input. The following example shows cat working without an argument; the **<** symbol causes the shell to redirect standard input to cat to come from **practice**. The shell passes no arguments to cat.

```
$ cat < practice
This is a small file that I created
using:  a text editor!

End.
```

You can use cat to create short, simple files. See "The Keyboard and Screen as Standard Input and Standard Output" on page 160 and "Redirection" on page 161 for more examples of redirecting the standard input and standard output of cat.

### OPTIONS

Number The **–n** (**––number**) option numbers all lines as they are written to standard output while the **–b** (**––number-nonblank**) numbers only non-blank lines.

```
$ cat -n practice
     1  This is a small file that I created
     2  using:  a text editor!
     3
     4  End.
```

Tabs The **–T** (**––show-tabs**) option displays TABs as **^I**.

```
$ cat -T practice
This is a small file that I created
using:^Ia text editor!

End.
```

You can combine options like this

```
$ cat -bT practice
```

Or, using long options, like this

```
$ cat --number-nonblank --show-tabs practice
```

Nonprinting The **–v** (**––show-nonprinting**) option displays CONTROL characters using the caret notation (e.g., **^M** means CONTROL-M) and displays characters that have the high bit set (META characters) using the **M-** notation (e.g., **M-M** means META-M). This option does not convert TABs and LINEFEEDs. Use **–T** (**––show-tabs**) if you want to display TABs as **^I**. LINEFEEDs cannot be displayed as anything but themselves; otherwise, the line could be too long.

### RELATED UTILITIES

tac The tac (cat spelled backwards) utility works the same way as cat works except it reverses the order of the lines in each file.

rev The rev (reverse) utility works the same way as cat works except it reverses the order of the characters in each line.

## LPI date: DISPLAYS THE SYSTEM TIME AND DATE

Without any arguments, the date utility displays the date and time known to the local system. If you set up a locale database (page 374), date uses that database to substitute in its output terms appropriate to the locale for your account. Use timedatectl (page 579) to set the system clock.

```
$ date
Mon Sep 29 18:02:26 PDT 2014
```

### The hardware clock and the system clock

tip The hardware clock is the time kept in the BIOS (page 32). This clock is battery powered and keeps time even when the computer is off. The system clock runs only while the system is running and is what Linux uses when it needs to know the time. On some systems the system clock is adjusted periodically by NTP (Network Time Protocol; page 321) so it stays accurate.

### ARGUMENTS

The following example formats and specifies the contents of the output of date. See the date man page for a complete list of format sequences.

```
$ date +"%A %B %d"
Monday September 29
```

### OPTIONS

Date The **–d** ***datestring*** (**––date=*datestring***) option displays the date specified by ***datestring***, not the date known to the system. According to the date man page, "the ***datestring*** is a mostly free-format date string" such as **2pm next thursday**. See **DATE STRING** in the date man page for details about the syntax of ***datestring***. This option does not change the system clock.

UTC The **–u** (**––utc** *or* **––universal**) option displays or sets the time and date using UTC (Universal Coordinated Time; page 1280). UTC is also called GMT (Greenwich Mean Time).

```
$ date -u
Tue Sep 30 00:50:18 UTC 2014
```

### RELATED UTILITIES

timedatectl The timedatectl utility displays more information about the system clock. You can also use it to set the system clock. See page 579.

cal The cal utility displays a calendar for the month and year you specify. Without any arguments it displays a calendar for the current month.

## LE+ echo: DISPLAYS ARGUMENTS

The echo utility copies its arguments, followed by a NEWLINE, to standard output. The Bourne Again Shell has an echo builtin that works similarly to the echo utility.

The echo builtin/utility is a good tool for learning about the shell and other Linux utilities. Some examples on page 174 use echo to illustrate how special characters, such as the asterisk, work. Throughout Chapters 5, 9, and 28, echo helps explain how shell variables work and how you can send messages from shell scripts to the screen. You can even use echo to create a short file by redirecting its output:

```
$ echo "This is a short file." > short
$ cat short
This is a short file.
```

### ARGUMENTS

The arguments can include quoted strings, ambiguous file references, and shell variables. The shell recognizes and expands unquoted special characters in arguments.

The following example shows echo copying its arguments to standard output. The second command includes an unquoted asterisk (*; page 174) that the shell expands into a list of files in the working directory before passing that list as arguments to echo; echo copies this list to standard output.

```
$ echo This is a sentence.
This is a sentence.
$ echo star: *
star: memo memo.0714 practice
$ ls
memo  memo.0714  practice
```

### OPTION

NEWLINE The **–n** option suppresses the NEWLINE that normally terminates the output of echo. This option is useful in shell scripts when you want to prompt a user and have the response appear on the same line as the prompt. See page 1063 for an example of a shell script that uses this feature.

## LPI hostname: DISPLAYS THE SYSTEM NAME

The hostname utility displays the name of the system you are working on. Use this utility if you are not sure that you are logged in on the correct machine. See also **/etc/hostname** on page 485.

```
$ hostname
guava
```

### OPTION

The following option works with hostname.

FQDN The **–f** (**––fqdn**) option displays the *FQDN* (page 1248) of the local system.

### RELATED UTILITY

hostnamectl The hostnamectl utility displays more information about the local system. You can also use it to change the hostname.

## LE less Is more: DISPLAY A TEXT FILE ONE SCREEN AT A TIME

You can use the less or more utilities, called *pagers,* to view a long text file one screen at a time. Each of these utilities pauses after displaying a screen of text. You can then press the SPACE bar to display the next screen of text.

Although less and more are very similar, they have subtle differences. The less utility, for example, allows you to move backward while viewing a file in some situations where more does not. Whereas more must read an entire file before displaying anything, less does not have to and so starts more quickly than more when displaying a large file. At the end of the file less displays an **END** message and waits for you to press **q** before returning you to the shell. In contrast, more returns you directly to the shell. While using both utilities you can press **h** to display a Help screen that lists commands you can use while paging through a file. Give the command **less /etc/services** to experiment with paging through a file.

### ARGUMENTS

Both less and more take the names of files you want to view as arguments. If you do not specify an argument or if you specify a hyphen (**–**; less only) in place of a filename, less and more read from standard input.

### OPTIONS

The following options work with less.

Clear screen The **–c** (**––clear-screen**) option paints each new screen from the top down; by default less scrolls the display.

EOF The **–e** (**––quit-at-eof**) option causes less to exit when it reaches the second EOF (when you press SPACE while less is displaying **END** at the end of a file) so you do not need to type **q** to quit.

Truncate The **–S** (**––chop-long-lines**) option truncates lines wider than the screen. By default less wraps long lines.

No initialization The **–X** (**––no-init**) option prevents less from sending terminal initialization and deinitialization strings to the terminal. By default, less clears the screen when it finishes; this option causes the last page of what you were viewing to remain on the screen.

**optional** You can set the **LESS** environment variable (page 1054) in your **~/.bash_profile** file (page 336) to set options for less each time you call it and when it is called from another program such as man. For example, the following line in the **.bash_profile** file in your home directory will cause less to always run with the **–X** option.

```
export LESS='-X'
```

### RELATED UTILITY

most The most utility (part of the **most** package; see page 512 for installation instructions) is newer and more capable than less and more. See the most man page for details.

## LPI ls: DISPLAYS INFORMATION ABOUT FILES

The ls utility displays information about one or more files. It lists the information alphabetically by filename unless you use an option to change the order.

When you do not provide an argument, ls displays the names of the visible files (those with filenames that do not begin with a period; page 188) in the working directory.

### ARGUMENTS

The arguments are one or more pathnames of any ordinary, directory, or device files. The shell expands ambiguous file references (page 173) in the arguments.

When you specify an ordinary file as an argument, ls displays information about that one file. When the argument is a directory, ls displays the contents of the directory. It displays the name of the directory only when needed to avoid ambiguity, such as when the listing includes more than one directory.

```
$ ls memos.zach/130715.1
memos.zach/130715.1

$ ls memos.zach
130712.1  130714.2  130714.3  130715.1

$ ls memos.*
memos.max:
130619.1  130621.2  130622.1

memos.sam:
130811.2  130811.3  130812.1

memos.zach:
130712.1  130714.2  130714.3  130715.1
```



### LE OPTIONS

Options determine the type of information ls displays, and the manner and order in which it displays the information. When you do not use an option, ls displays a short list that contains just the names of files, in alphabetical order. See page 153 for examples of how options affect ls output.

All The **–a** (**––all**) option includes hidden filenames (those filenames that begin with a period; page 188) in the listing. Without this option ls does not list information about files with hidden filenames unless you specify the name of a hidden file as an argument. The * ambiguous file reference does not match a leading period in a filename, so you must use this option or explicitly specify a filename (ambiguous or not) that begins with a period to display information about files with hidden filenames.

LE Directory The **–d** (**––directory**) option displays directories without displaying their contents.

```
$ ls -ld /
dr-xr-xr-x. 27 root root 4096 07-16 18:37 /
```

This option is useful when you want to find out, for example, the name of the user who owns a directory in a large directory such as **/etc**. Instead of scrolling through the output of **ls –l /etc** looking for a directory, you can give the command **ls –ld /etc/*filename*** (e.g., **ls –ld /etc/cron.d**).

Human readable The **–h** (**––human-readable**) option, when specified with the **–l** option, displays sizes in K (kilobyte), M (megabyte), and G (gigabyte) blocks, as appropriate. This option works with the **–l** and **–s** options only. It displays powers of 1,024. Use **––si** to display powers of 1,000.

```
$ ls -lh /bin
...
-rwxr-xr-x 1 root root  15K 06-03 13:54 tailf
-rwxr-xr-x 1 root root 346K 2014-02-04  tar
-rwxr-xr-x 1 root root  11K 2014-08-27  tempfile
-rwxr-xr-x 1 root root  59K 2014-03-24  touch
-rwxr-xr-x 1 root root  27K 2014-03-24  true
...
```

Long The **–l** (lowercase "el"; **––format=long**) option lists more information about each file. See page 199 for an explanation of this output. If standard output for a directory listing is sent to the screen, this option displays the number of blocks used by all files in the listing on a line before the listing.

LE Recursive The **–R** (**––recursive**) option recursively lists directory hierarchies.

Reverse The **–r** (**––reverse**) option displays the list of filenames in reverse sorted order.

LE Size The **–s** (**––size**) option displays the number of 1,024-byte blocks allocated to the file. The size precedes the filename. With the **–l** option, this option displays the size in column 1 and shifts other items one column to the right. If standard output for a directory listing is sent to the screen, this option displays the number of blocks used by all files in the listing on a line before the listing. You can include the **–h** option to make the file sizes easier to read. The following example recursively lists the **memos** directory hierarchy in reverse size order.

```
$ ls -lRrs memos
memos:
4 drwxrwxr-x. 2 sam sam 4096 04-29 14:11 memos.zach
4 drwxrwxr-x. 2 sam sam 4096 04-29 14:32 memos.sam
4 drwxrwxr-x. 2 sam sam 4096 04-30 14:15 memos.max

memos/memos.zach:
4 -rw-rw-r--. 1 sam sam 4064 04-29 14:11 130715.1
4 -rw-rw-r--. 1 sam sam 3933 04-29 14:11 130714.3
4 -rw-rw-r--. 1 sam sam 3317 04-29 14:11 130714.2
...
```

## LE+ rm: REMOVES A FILE (DELETES A LINK)

The rm utility removes hard and/or symbolic links to one or more files. Frequently this action results in the file being deleted. See page 211 for information on links.

### ARGUMENTS

The arguments are the pathnames of the files whose links rm will remove. Removing the only (last) hard link to a file deletes the file (page 216). Removing a symbolic link deletes the symbolic link only.

#### Be careful when you use rm with ambiguous file references

**caution** Because this utility enables you to remove a large number of files with a single command, use rm cautiously, especially when you are working with ambiguous file references. *Never use* rm *with ambiguous file references while you are working with* **root** *privileges.* If you have any doubts about the effect of an rm command with an ambiguous file reference, first use echo with the same file reference and evaluate the list of files the reference generates. Alternately, you can use the rm **–i** (**––interactive**) option.

### OPTIONS

Force The **–f** (**––force**) option, without asking for your consent, removes files for which you do not have write access permission. This option suppresses informative messages if a file does not exist.

Interactive The **–i** (**––interactive**) option prompts you before removing each file. If you use **–r** (**––recursive**) with this option, rm also prompts you before examining each directory.

```
$ rm -ri memos
rm: descend into directory 'memos'? y
rm: descend into directory 'memos/memos.max'? y
rm: remove regular file 'memos/memos.max/130621.2'? y
rm: remove regular file 'memos/memos.max/130619.1'? n
...
```

You can create an alias (page 398) for **rm –i** and put it in a startup file (page 188) so rm always runs in interactive mode.

Recursive The **–r** (**––recursive**) option deletes the contents of the specified directory, including all its subdirectories, and the directory itself. Use this option with caution; do not use it with wildcards (* and ?).

Verbose The **–v** (**––verbose**) option displays the name of each file as it is removed.

### RELATED UTILITIES

testdisk The testdisk utility (**testdisk** package) scans, repairs, and sometimes can recover disk partitions. In some cases it can undelete files. See the testdisk man page for details.

# WORKING WITH FILES

This section describes utilities that copy, move, print, search through, display, sort, compare, and identify files.

### Filename completion

**tip** After you enter one or more letters of a filename (as an argument) on a command line, press TAB, and the shell will complete as much of the filename as it can. When only one filename starts with the characters you entered, the shell completes the filename and places a SPACE after it. You can keep typing or you can press RETURN to execute the command at this point. When the characters you entered do not uniquely identify a filename, the shell completes what it can and waits for more input. If pressing TAB does not change the display, press TAB again to display a list of possible completions. For more information refer to "Pathname Completion" on page 395.

## LE+ cp: COPIES FILES

The cp utility copies one or more files. Use scp (page 723) or rsync (page 724) to copy files from one system to another (or to make local copies). See page 245 for a description of the related mv (move) utility.



### ARGUMENTS

With two arguments that are pathnames, neither of which specifies a directory, cp copies the file named by the first argument to the file named by the second argument.

```
$ cp memo memo.cp
```

With two or more arguments that are pathnames, the last of which is an existing directory, cp copies the files named by all but the last argument to the directory named by the last argument.

```
$ cp memo1 memo2 memo4 memo.dir
```

### cp can destroy a file

**caution** If the destination file exists *before* you give a cp command, cp overwrites it. Because cp overwrites (and destroys the contents of) an existing destination file without warning, you must take care not to cause cp to overwrite a file that you need. The cp **–i** (interactive) option prompts you before it overwrites a file.

The following example assumes the file named **orange.2** exists before you give the cp command. The user answers **y** to overwrite the file.

```
$ cp -i orange orange.2
cp: overwrite 'orange.2'? y
```

### OPTIONS

Archive The **–a** (**––archive**) option attempts to preserve the owner, group, permissions, access date, and modification date of source file(s) while copying a directory recursively.

Backup The **–b** (**––backup**) option makes a backup copy of a file that would be removed or overwritten by cp. The backup copy has the same name as the destination file with a

tilde (~) appended to it. When you use both **–b** and **–f,** cp makes a backup copy when you try to copy a file over itself. For more backup options, search for **Backup options** in the **coreutils** info page.

Force The **–f** (**––force**) option causes cp to try to remove the destination file (when it exists but cannot be opened for writing) before copying the source file. This option is useful when the user copying a file does not have write permission to an existing file but does have write permission to the directory containing the file. Use this option with **–b** to back up a destination file before removing or overwriting it.

Interactive The **–i** (**––interactive**) option prompts you whenever cp would overwrite a file. If you respond with a string that starts with **y** or **Y**, cp copies the file. If you enter anything else, cp does not copy the file.

```
$ cp -i * ../memos.helen
cp: overwrite '../memos.helen/130619.1'? y
cp: overwrite '../memos.helen/130622.1'? n
```

Preserve The **–p** (**––preserve**[=***attr***]) option creates a destination file with the same owner, group, permissions, access date, modification date, and ACLs as the source file. The **–p** option does not take an argument.

Without ***attr***, **––preserve** works as described above. The ***attr*** is a comma-separated list that can include **mode** (permissions), **ownership** (owner and group), **timestamps** (access and modification dates), **links** (hard links), and **all** (all attributes).

Recursive The **–R** *or* **–r** (**––recursive**) option recursively copies directory hierarchies including ordinary files. The last argument must be the name of a directory.

Verbose The **–v** (**––verbose**) option displays the name of each file as cp copies it.

```
$ cp -rv memos memos.bak
'memos' -> 'memos.bak'
'memos/memos.max' -> 'memos.bak/memos.max'
'memos/memos.max/130619.1' -> 'memos.bak/memos.max/130619.1'
'memos/memos.max/130622.1' -> 'memos.bak/memos.max/130622.1'
'memos/memos.sam' -> 'memos.bak/memos.sam'
'memos/memos.sam/130811.3' -> 'memos.bak/memos.sam/130811.3'
'memos/memos.sam/130811.2' -> 'memos.bak/memos.sam/130811.2'
...
```

## LE+ cut: SELECTS CHARACTERS OR FIELDS FROM INPUT LINES

The cut utility selects characters or fields from lines of input and writes them to standard output. Character and field numbering start with 1. Although limited in functionality, cut is easy to learn and use and is a good choice when columns and fields can be specified without using pattern matching.

### ARGUMENTS

Arguments to cut are pathnames of ordinary files. If you do not specify an argument or if you specify a hyphen (–) in place of a pathname, cut reads from standard input.

### OPTIONS

Characters

The **–c** ***clist*** (**––characters=*clist***) option selects the characters given by the column numbers in ***clist***. The value of ***clist*** is one or more comma-separated column numbers or column ranges. A range is specified by two column numbers separated by a hyphen. A range of ***–n*** means columns **1** through ***n***; ***n–*** means columns ***n*** through the end of the line.

The following example displays the permissions of the files in the working directory. The **–c2-10** option selects characters 2 through 10 from each input line.

```
$ ls -l | cut -c2-10
otal 2944
rwxr-xr-x
rw-rw-r--
rw-rw-r--
rw-rw-r--
rw-rw-r--
```

Input delimiter

The **–d** ***dchar*** (**––delimiter=*dchar***) option specifies ***dchar*** as the input field delimiter. This option also specifies ***dchar*** as the output field delimiter unless you use the **––output-delimiter** option. The default delimiter is a TAB character. Quote ***dchar*** as necessary to protect it from shell expansion.

Fields

The **–f** ***flist*** (**––fields=*flist***) option selects the fields specified by ***flist***. The value of ***flist*** is one or more comma-separated field numbers or field ranges. A range is specified by two field numbers separated by a hyphen. A range of ***–n*** means fields **1** through ***n***; ***n–*** means fields ***n*** through the last field. The field delimiter is a TAB character unless you use the **–d** option to change it.

The following example displays a list of full names as stored in the fifth field (**–f5**) of the **/etc/passwd** file. The **–d** option specifies that the colon character is the field delimiter.

```
$ cut -d: -f5 /etc/passwd
Sam the Great
Sam the Great
Zach Brill
Max Wild
...
```

The next command outputs the size and name of each file in the working directory. The **–f** option selects the fifth and ninth fields from the input lines. The **–d** option tells cut to use SPACEs, not TABs, as delimiters. The tr utility (page 266) with the **–s** option changes sequences of two or more SPACE characters to a single SPACE; otherwise, cut counts the extra SPACE characters as separate fields.

```
$ ls -l | tr -s '  ' ' ' | cut -f5,9 -d' '

259 countout
9453 headers
1474828 memo
1474828 memos_save
```

```
7134 tmp1
4770 tmp2
13580 typescript
```

Output delimiter

The **––output-delimiter=***ochar* option specifies *ochar* as the output field delimiter. By default, the output field delimiter is the same as the input field delimiter (the TAB character unless you change it using **–d**). Quote *ochar* as necessary to protect it from shell expansion.

## diff: DISPLAYS THE DIFFERENCES BETWEEN TWO TEXT FILES

The diff (difference) utility displays line-by-line differences between two text files. By default diff displays the differences as instructions that you can use to edit one of the files to make it the same as the other.

The sdiff utility is similar to diff but its output might be easier to read; its output is the same as that of the diff **–y** (**––side-by-side**) option. Use the diff3 utility to compare three files and cmp to compare nontext (binary) files.

### ARGUMENTS

The diff utility commonly takes the pathnames of two ordinary files it is to compare as arguments.

When you call diff without any options, it produces a series of lines containing Add (**a**), Delete (**d**), and Change (**c**) instructions. Each of these lines is followed by the lines from the file you need to add to, delete from, or change, respectively, to make the files the same. A less than symbol (**<**) precedes lines from ***file1***. A greater than symbol (**>**) precedes lines from ***file2***. The diff output appears in the format shown in Table 7-1. A pair of line numbers separated by a comma represents a range of lines; a single line number represents a single line.

**Table 7-1** diff output

| Instruction | Meaning (to change file1 to file2) |
|---|---|
| *line1* a *line2*,*line3*<br>> lines from file2 | Append *line2* through *line3* from **file2** after *line1* in **file1** |
| *line1*,*line2* d *line3*<br>< lines from file1 | Delete *line1* through *line2* from **file1** |
| *line1*,*line2* c *line3*,*line4*<br>< lines from file1<br>---<br>> lines from file 2 | Change *line1* through *line2* in **file1** to *line3* through *line4* from **file2** |

The diff utility assumes you will convert the file named by the first argument to the file named by the second argument. The line numbers to the left of each of the **a**, **c**, or **d** instructions always pertain to the first file; the line numbers to the right of the instructions apply to the second file. To display instructions to convert the files in the opposite order, reverse the arguments.

BLANK

# 10

# System Administration: Core Concepts



## In This Chapter



## Objectives

After reading this chapter you should be able to:

- Describe the startup sequence using Upstart
- Manage which services start at boot time
- List four characteristics of a well-maintained system
- Start and stop services on a running system
- Boot into single-user mode for system maintenance
- Shut down a running system
- Use system administration tools to monitor and maintain the system
- List common steps for installing, configuring, and securing a server
- Configure a system using a static IP address or using DHCP
- Explain the purpose of the GRUB boot loader
- Modify the boot loader configuration
- Troubleshoot kernel messages using `dmesg`

Administrator The job of a system administrator is to keep one or more systems in a useful and convenient state for users. On a Linux system, the administrator and user might both be you, with you and the computer being separated by only a few feet. Alternately, the system administrator might be halfway around the world, supporting a network of systems, with you being one of thousands of users. Or a system administrator can be one person who works part-time taking care of a system and perhaps is also a user of the system. In some cases several administrators might work together full-time to keep many systems running. See page 596 for a technical description of an administrator.

A well-maintained system:

- Runs quickly enough so users do not get frustrated waiting for the system to respond or complete a task
- Has enough storage to accommodate the reasonable needs of users
- Provides a working environment appropriate to each user's abilities and requirements
- Is secure from malicious and accidental acts altering its performance or compromising the security of the data it holds and exchanges with other systems
- Is backed up regularly, with recently backed-up files readily available to users
- Has recent copies of the software that users need to get their jobs done
- Is easier to administer than a poorly maintained system

In addition, a system administrator should be available to help users with all types of system-related problems—from logging in to obtaining and installing software updates to tracking down and fixing obscure network issues.



Part III of this book breaks system administration into nine chapters.

- Chapter 9 covers bash (Bourne Again Shell) to a depth that you can use it interactively to administer a system and begin to understand complex administration shell scripts.
- Chapter 10 (this chapter) covers the core concepts of system administration, including working with **root** privileges, system operation, configuration tools and other useful utilities, as well as general information about setting up and securing a server (including a section on DHCP).
- Chapter 11 covers files, directories, and filesystems from an administrator's point of view.
- Chapter 12 covers installing software on the system, including the use of APT (apt-get), the Debian package (**dpkg**) management system, BitTorrent, and curl.
- Chapter 13 discusses how to set up local and remote printers that use the CUPS printing system.

- Chapter 14 covers additional system administrator tasks and tools, including setting up users and groups, backing up files, scheduling tasks, printing system reports, and general problem solving.
- Chapter 15 discusses system security including su, sudo, passwords, cryptography, SSL certificates, hashes, GPG, and PAM.
- Chapter 16 goes into detail about how to set up a LAN, including setting up and configuring network hardware and configuring software.
- Chapter 17 describes how to set up virtual machines locally (gnome-boxes, KVM/QEMU, and VMware) and in the cloud (AWS).

Because Linux is readily configurable and runs on a wide variety of platforms, this chapter cannot discuss every system configuration or every action you might have to take as a system administrator. Instead, this chapter seeks to familiarize you with the concepts you need to understand and the tools you will use to maintain a Linux system. Where it is not possible to go into depth about a subject, the chapter provides references to other sources.

This chapter assumes you are familiar with the following terms:

*block device* (page 1236)
*daemon* (page 1243)
*device* (page 1244)
*device filename* (page 1244)
*disk partition* (page 1244)
*environment* (page 1246)
*filesystem* (page 1247)
*fork* (page 1248)
*kernel* (page 1255)
*login shell* (page 1257)
*mount* (page 1260)
*process* (page 1266)
*root filesystem* (page 1270)
*runlevel* (page 1271)
*signal* (page 1272)
*spawn* (page 1274)
*system console* (page 1276)
*X server* (page 1282)



## LPI THE UPSTART EVENT-BASED init DAEMON

Because the traditional System V **init** daemon (SysVinit) does not deal well with modern hardware, including hotplug (page 495) devices, USB hard and flash drives, SD cards, and network-mounted filesystems, Ubuntu replaced it with the Upstart **init** daemon (upstart.ubuntu.com and upstart.ubuntu.com/cookbook).

Several other replacements for SysVinit are also available. One of the most prominent is **systemd** (www.freedesktop.org/wiki/Software/systemd, fedoraproject.org/wiki/Systemd, and www.markshuttleworth.com/archives/1316). The **systemd** init daemon is used by Fedora and Red Hat Enterprise Linux. And, starting with release 14.10, will be used by Ubuntu. Solaris uses SMF (Service Management Facility) and MacOS uses **launchd**.

The runlevel-based SysVinit daemon uses runlevels (recovery/single-user, multiuser, and more) and links from the **/etc/rc?.d** directories to the init scripts in **/etc/init.d** to start and stop system services (page 435). The event-based Upstart **init** daemon uses events to start and stop system services. With the 6.10 release (Edgy), Ubuntu

switched to the Upstart **init** daemon and began making the transition from the SysVinit setup to the Upstart setup. This section discusses Upstart and the parts of SysVinit that remain: the **/etc/rc?.d** and **/etc/init.d** directories and the concept of runlevels.

The Upstart **init** daemon is event based and runs specified programs when something on the system changes. These programs, which are frequently scripts, start and stop services. This setup is similar in concept to the links to init scripts that SysVinit calls as a system enters runlevels, except Upstart is more flexible. Instead of starting and stopping services only when the runlevel changes, Upstart can start and stop services upon receiving information that something on the system has changed. Such a change is called an *event*. For example, Upstart can take action when it learns from udev (page 494) that a filesystem, printer, or other device has been added or removed from the running system. It can also start and stop services when the system boots, when the system is shut down, or when a job changes state.

Ubuntu has moved away from the SysVinit setup to the cleaner, more flexible Upstart setup. Because most system services have been put under the control of Upstart, entries in the **/etc/init** directory have replaced the contents of the **/etc/init.d** and **/etc/rc?.d** directories. Runlevels are no longer a formal feature of Ubuntu, although they are maintained for compatibility with third-party software.



## SOFTWARE PACKAGE

The Upstart system is contained in one package, which is installed by default.

- **upstart**—Provides the Upstart **init** daemon and initctl utility.

## TERMINOLOGY

Event An *event* is a change in state that can be communicated to **init**. Almost any change in state—either internal or external to the system—can trigger an event. For example, the boot loader triggers the **startup** event (**startup** man page) and the telinit command (page 438) triggers the **runlevel** event (page 438). Removing and installing a hotplug (page 495) or USB device (such as a printer) can trigger an event as well. You can also trigger an event manually by giving the initctl **emit** command (page 432). For more information refer to "Events" on page 432.

Job A *job* is a series of instructions that **init** reads. These instructions typically include a program (binary file or shell script) and the name of an event. The Upstart **init** daemon runs the program when the event is triggered. You can run and stop a job manually by giving the initctl **start** and **stop** commands, respectively (page 432). Jobs are divided into tasks and services. A job is a service by default; you must explicitly specify a job as a task for it to run as a task.

Task A *task* is a job that performs its work and returns to a waiting state when it is done. A task blocks the program/process that emitted the event that triggered it until the program it specifies is finished running. The **rc** task described on page 433 is an example of a task.

Service

A *service* is a job that does not normally terminate by itself. For example, the **logd** daemon and getty processes (page 434) are implemented as services. The **init** daemon monitors each service, restarting the service if it fails and killing the service if it is stopped either manually or by an event. A service blocks the program/process that emitted the event that triggered it until the program it specifies has started running.

Job definition file

The **/etc/init** directory holds *job definition files* (files defining the jobs that the Upstart **init** daemon runs). Initially this directory is populated by the Upstart software package. The installation of some services (servers) adds files to this directory to control the service, replacing the files that were previously placed in the **/etc/rc?.d** and **/etc/init.d** directories when the service was installed.

**init** is a state machine

At its core, the Upstart **init** daemon is a state machine. It keeps track of the state of jobs and, as events are triggered, tracks jobs as they change states. When **init** tracks a job from one state to another, it may execute the job's commands or terminate the job.

Runlevel emulation

The System V **init** daemon used changes in runlevels (page 438) to determine when to start and stop processes. Ubuntu systems, which rely on the Upstart **init** daemon, have no concept of runlevels. To ease migration from a runlevel-based system to an event-based system, and to provide compatibility with software intended for other distributions, Ubuntu emulates runlevels using Upstart.

The **rc** task, which is defined by the **/etc/init/rc.conf** file, runs the **/etc/init.d/rc** script. This script, in turn, runs the **init** scripts in **/etc/init.d** from the links in the **/etc/rc?.d** directories, emulating the functionality of these links under SysVinit. The **rc** task runs these scripts as the system enters a runlevel; it normally takes no action when the system leaves a runlevel. See page 433 for a discussion of the **rc** task and page 435 for information on init scripts. Upstart implements the runlevel (page 438) and telinit (page 438) utilities to provide compatibility with SysVinit systems.

initctl

The initctl (**init** control) utility communicates with the Upstart **init** daemon. An ordinary user can query the Upstart **init** daemon by using the initctl **list** and **status** commands. A system administrator working with **root** privileges can both query this daemon and start and stop jobs. For example, the initctl **list** command lists jobs and their states:

```
$ initctl list
avahi-daemon start/running, process 571
mountnfs-bootclean.sh start/running
rsyslog start/running, process 544
tty4 start/running, process 903
udev start/running, process 347
upstart-udev-bridge start/running, process 338
whoopsie start/running, process 6162
...
```

See the initctl man page and the examples in this section for more information. You can give the command **initctl help** (no hyphens before **help**) to display a list of initctl

commands. Alternately, you can give commands similar to the following one to display more information about an initctl command:

```
$ initctl list --help
Usage: initctl list [OPTION]...
List known jobs.

Options:
      --session               use existing D-Bus session bus to connect to
                                init daemon (for testing)
      --system                use D-Bus system bus to connect to init daemon
      --dest=NAME             destination well-known name on D-Bus bus
      --user                  run in user mode (as used for user sessions)
  -q, --quiet                 reduce output to errors only
  -v, --verbose               increase output to include informational messages
      --help                  display this help and exit
      --version               output version information and exit

The known jobs and their current status will be output.

Report bugs to <upstart-devel@lists.ubuntu.com>
```

Replace **list** with the initctl command for which you want to obtain more information. The start, stop, reload, restart, and status utilities are links to initctl that run the initctl commands they are named for. As explained in the next section, many administrators use the service utility to call initctl to change the status of servers.

# JOBS

Each file in the **/etc/init** directory defines a job and usually contains at least an event and a command. When the event is triggered, **init** executes the command. This section describes server jobs (daemons [services] installed with servers), administrator-defined jobs, and jobs installed with the **upstart** package.

## LPI SERVER/SERVICE (DAEMON) JOBS

Most servers, also called services (e.g., Apache, Samba), can be controlled using initctl commands. You can start a server using start (a link to initctl that runs the initctl **start** command) and stop (a link to initctl that runs the initctl **stop** command). The restart, reload, and status utilities are also links to initctl. The restart utility issues initctl **stop** and then initctl **start** commands; reload sends a SIGHUP signal to a job, usually causing it to reread its configuration files before restarting; and **status** displays status information. For example

```
$ sudo stop cups
cups stop/waiting

$ sudo start cups
cups start/running, process 2485

$ status cups
cups start/running, process 2485
```

LPI service However, for consistency this book uses the service utility to call initctl to change the status of servers. Using service, preceding commands appear as

```
$ sudo service cups stop
cups stop/waiting

$ sudo service cups start
cups start/running, process 2485

$ service cups status
cups start/running, process 2485
```

## ADMINISTRATOR-DEFINED JOBS

**mudat** example The following administrator-defined job uses the **exec** keyword to execute a shell command. You can also use this keyword to execute a shell script stored in a file or a binary executable file.

In the first stanza (**start on runlevel 2**), **start on** is a keyword (you can use **stop on** in its place), **runlevel** is an event (page 428), and **2** is an argument to **runlevel**.

```
$ cat /etc/init/mudat.conf
start on runlevel 2
task
exec echo "Entering multiuser mode on" $(date) > /tmp/mudat.out
```

This file defines a job: It runs the echo shell builtin when the system enters multiuser mode (runlevel 2). This command writes a message that includes the time and date to **/tmp/mudat.out**. The shell uses command substitution (page 416) to execute the date utility. After this job runs to completion, the **mudat** task stops and enters a wait state.

In the next example, the cat utility shows the contents of the **/tmp/mudat.out** file and the initctl **list** command and the status utility report on this task:

```
$ cat /tmp/mudat.out
Entering multiuser mode on Wed Jun 25 16:49:10 PDT 2014

$ initctl list | grep mudat
mudat stop/waiting
$ status mudat
mudat stop/waiting
```

If the **exec** command line contains shell special characters (page 150), **init** executes **/bin/sh** (a link to dash [page 335]) and passes the command line to the shell. Otherwise, **exec** executes the command line directly. To run multiple shell commands, either use **exec** to run a shell script stored in a file or use **script...end script** (discussed next).

**myjob** example You can also define an event and set up a job that is triggered by that event. The **myjob.conf** job definition file defines a job that is triggered by the **hithere** event:

```
$ cat /etc/init/myjob.conf
start on hithere
script
    echo "Hi there, here I am!" > /tmp/myjob.out
    date >> /tmp/myjob.out
end script
```

The **myjob** file shows another way of executing commands: It includes two command lines between the **script** and **end script** keywords. These keywords always cause **init** to execute **/bin/sh**. The commands write a message and the date to the **/tmp/myjob.out** file. You can use the initctl **emit** command to trigger the job.

initctl **emit**

```
$ sudo initctl emit hithere

$ cat /tmp/myjob.out
Hi there, here I am!
Tue Jun 24 14:33:38 PDT 2014

$ status myjob
myjob stop/waiting
```

initctl **start** and **stop** In the preceding example, cat shows the output that **myjob** generates and initctl displays the status of the job. You can run the same job by giving the command **initctl start myjob** (or just **start myjob**). The initctl **start** command is useful when you want to run a job without triggering an event. For example, you can use the command **initctl start mudat** to run the **mudat** job from the previous example without triggering the **runlevel** event.

## EVENTS

The **upstart** package defines many events. The following command lists events and brief descriptions of each. See the corresponding man page for more information on each event.

```
$ apropos event 2| grep signalling
all-swaps (7)         - event signalling that all swap partitions have been activated
control-alt-delete (7) - event signalling console press of Control-Alt-Delete
dbus-event (7)        - event signalling that a dbus signal has been emitted
file-event (7)        - event signalling that a file has changed
filesystem (7)        - event signalling that filesystems have been mounted
keyboard-request (7) - event signalling console press of Alt-UpArrow
local-filesystems (7) - event signalling that local filesystems have been mounted
mounted (7)           - event signalling that a filesystem has been mounted
mounting (7)          - event signalling that a filesystem is mounting
power-status-changed (7) - event signalling change of power status
remote-filesystems (7) - event signalling that remote filesystems have been mounted
runlevel (7)          - event signalling change of system runlevel
session-end (7)       - event signalling session shutdown
socket-event (7)      - event signalling that a socket connection has been made
started (7)           - event signalling that a job is running
starting (7)          - event signalling that a job is starting
startup (7)           - event signalling system startup
stopped (7)           - event signalling that a job has stopped
stopping (7)          - event signalling that a job is stopping
virtual-filesystems (7) - event signalling that virtual filesystems have been mounted
```

apropos The apropos utility (page 137), which is a link to whatis, sends its output to standard error. The **2|** operator is a pipeline (page 166) that sends standard error (page 339) of apropos to standard input of grep.

optional

### SPECIFYING EVENTS WITH ARGUMENTS

The telinit (page 438) and shutdown (page 441) utilities emit **runlevel** events that include arguments. For example, **shutdown** emits **runlevel 0**, and **telinit 2** emits **runlevel 2**. You can match these events within a job definition using the following syntax:

*start|stop on* ***event*** *[**arg** [**arg**...]]*

where ***event*** is an event such as **runlevel** and ***arg*** is one or more arguments. To stop a job when the system enters runlevel 2 from runlevel 1, specify **stop on runlevel 2 1**. You can also specify **[235]** to match 2, 3, and 5 or **[!2]** to match any value except 2.

Event arguments

Although Upstart ignores additional arguments in an event, additional arguments in an event name within a job definition file must exist in the event. For example, **runlevel** (no argument) in a job definition file matches all **runlevel** events (regardless of arguments), whereas **runlevel S arg1** does not match any **runlevel** event because the **runlevel** event takes two arguments (the runlevel the system is entering and the previous runlevel).

## JOB DEFINITION FILES IN /etc/init

This section describes some of the jobs that the **upstart** package puts in the **/etc/init** directory.

**rc** task and the **runlevel** event

A **runlevel** event [runlevel(7) man page] signals a change in runlevel and is emitted by telinit (page 438) and shutdown (page 441). This event sets the environment variable **RUNLEVEL** to the value of the new runlevel and sets **PREVLEVEL** to the value of the previous runlevel.

The **/etc/init/rc.conf** job definition file (next page) defines the **rc** task. This task monitors the **runlevel** event. The keyword **task** near the end of the file specifies this job as a task and not a service. Because it is a task, it blocks the call that emitted the **runlevel** event until the **rc** task has finished running.

The **exec** stanza in **rc.conf** calls the **/etc/init.d/rc** script (not the **rc** task) with an argument of **RUNLEVEL**. The **rc** script calls the links in the **/etc/rc*n*.d** directory, where ***n*** is equal to **RUNLEVEL** (page 435). Thus the **rc** task, when called with **RUNLEVEL** set to 2, runs the init scripts that the links in the **/etc/rc2.d** directory point to.

The **rc** task runs the **rc** script when the system enters a runlevel from 0 through 6 (**start on runlevel [0123456]**). Normally this task terminates when it finishes executing the **rc** script.

The stop stanza (**stop on runlevel [!$RUNLEVEL]**) takes care of the case wherein a second **runlevel** event attempts to start while an **rc** task is running the **rc** script. In this case, the value of **RUNLEVEL** is not equal to the value of **RUNLEVEL** that the **rc** task was called with and the **rc** task stops.

```
$ cat /etc/init/rc.conf
# rc - System V runlevel compatibility
#
# This task runs the old System V-style rc script when changing between
# runlevels.

description"System V runlevel compatibility"
author  "Scott James Remnant <scott@netsplit.com>"

emits deconfiguring-networking
emits unmounted-remote-filesystems

start on runlevel [0123456]
stop on runlevel [!$RUNLEVEL]

export RUNLEVEL
export PREVLEVEL

console output
env INIT_VERBOSE

task

script
if [ "$RUNLEVEL" = "0" -o "$RUNLEVEL" = "1" -o "$RUNLEVEL" = "6" ]; then
    status plymouth-shutdown 2>/dev/null >/dev/null && start wait-for
state WAITER=rc WAIT_FOR=plymouth-shutdown || :
fi
/etc/init.d/rc $RUNLEVEL
end script
```

tty services

Following is the job definition file for the service that starts and monitors the getty process (page 440) on **tty1**:

```
$ cat /etc/init/tty1.conf
# tty1 - getty
#
# This service maintains a getty on tty1 from the point the system is
# started until it is shut down again.

start on stopped rc RUNLEVEL=[2345] and (
            not-container or
            container CONTAINER=lxc or
            container CONTAINER=lxc-libvirt)

stop on runlevel [!2345]

respawn
exec /sbin/getty -8 38400 tty1
```

The event in the **start on** stanza is named **stopped** (see the **stopped** man page). This stanza starts the **tty1** service when the **rc** task is stopped with **RUNLEVEL** equal to 2, 3, 4, or 5. (The lines including **container** add conditions relating to containers.) Because the **rc** task is stopped as the system finishes entering each of these runlevels, the **tty1** service starts when the system enters any of these runlevels.

The event in the **stop on** stanza is named **runlevel**. This stanza stops the **tty1** service when a **runlevel** event is emitted with an argument other than 2, 3, 4, or 5—that is, when the system enters recovery mode, is shut down, or is rebooted.

The **respawn** keyword tells **init** to restart the **tty1** service if it terminates. The **exec** stanza runs a getty process with no parity (–8) on tty1 at 38,400 baud. In the next example, the initctl utility reports that the **tty1** service has started and is running as process 1031; ps reports on the process:

```
$ status tty1
tty1 start/running, process 1031
$ ps -ef | grep 1031
root      1031     1  0 13:26 tty1     00:00:00 /sbin/getty -8 38400 tty1
```

**control-alt-delete** task

See page 442 for a discussion of the **control-alt-delete** task, which you can use to bring the system down.

*LPI* **rc-sysinit** task and **inittab**

Under SysVinit, the **initdefault** entry in the **/etc/inittab** file tells **init** which runlevel (page 438) to bring the system to when it comes up. Ubuntu does not include an **inittab** file; instead, by default, the Upstart **init** daemon (using the **rc-sysinit** task) boots the system to multiuser mode (runlevel 2). If you want the system to boot to a different runlevel, modify the following line in the **rc-sysinit.conf** file:

```
$ cat /etc/init/rc-sysinit.conf
...
env DEFAULT_RUNLEVEL=2
...
```

### Do not set the system to boot to runlevel 0 or 6

**caution** Never set the system to boot to runlevel 0 or 6, as the system will not come up properly. To boot to multiuser mode (runlevel 2), set **DEFAULT_RUNLEVEL** to **2**. To boot to recovery mode, set **DEFAULT_RUNLEVEL** to **S**.

Changing the value of **DEFAULT_RUNLEVEL** from **2** to **S** causes the system to boot to recovery mode (runlevel S). If the **root** account is locked (as it is when Ubuntu is installed; page 613), after a few moments bash displays the **root** prompt (#). If the **root** account is unlocked, you will have to enter the **root** password before continuing.

**optional**

## *LPI* SysVinit (rc) Scripts: Start and Stop System Services

rc scripts

The SysVinit daemon is driven by init (initialization) scripts. These scripts, also called rc (run command) scripts, are shell scripts located in the **/etc/init.d** directory. The scripts are run via symbolic links in the **/etc/rc*n*.d** directories, where ***n*** is the runlevel the system is entering. Today, Ubuntu runs these scripts using Upstart.

### Most of the files in the /etc/rc*n*.d and /etc/init.d directories are gone

**tip** Under Upstart, Linux uses targets with the names of runlevels to aid migration and provide compatibility with software for other distributions (see Table 10-1 on page 438). Most of the files in the **/etc/rc*n*.d** and **/etc/init.d** directories have gone away; they have been replaced by Upstart job files.

BLANK

# 15

# System Security



## In This Appendix



## Objectives

After reading this chapter you should be able to:

- Explain what makes a user an administrator
- Describe what **root** privileges allow you to do
- Explain the need for and the responsibility of a privileged user (**root**)
- Gain **root** privileges using su and sudo
- Explain how a chroot jail works
- In general terms, explain how symmetric key encryption and asymmetric key encryption work
- Describe how hybrid encryption works and why you would use it
- List three characteristics of a hash function
- Describe how a man-in-the-middle attack works
- Explain how a salt protects against a dictionary attack
- Use **gpg --genkey** to generate a GPG key pair
- Find the author's public key on a keyserver

# LE RUNNING COMMANDS WITH root PRIVILEGES

Some commands can damage the filesystem or crash the operating system. Other commands can invade users' privacy or make the system less secure. To keep a Linux system up and running as well as secure, most systems are configured not to permit ordinary users to execute some commands or to access certain files. Linux allows a trusted user to execute these commands and access these files. A user running with these privileges is referred to as a *system administrator,* a *privileged user,* or *Superuser*.

## ADMINISTRATOR

The *system administrator,* or simply *administrator,* is a user who is permitted to run programs with **root** privileges. While working with **root** privileges, this user has extraordinary systemwide powers and the ability to run any program and examine any file. The administrator is discussed and referenced throughout this book. The role of the administrator is to keep one or more systems in a useful and convenient state for users (page 426). Because the administrator is ultimately responsible for system security, this summary section appears in this chapter. It points to (and is pointed to by) key discussions of the system administrator throughout this book.

**sudo** group As the system is set up, a user who is a member of the **sudo** group (page 98) can use sudo to gain **root** privileges (next); the first user you set up is automatically made a member of the **sudo** group. When you add or modify a user, you can specify that the user is an administrator by making the user a member of the **sudo** group (page 604).

No **root** password As Ubuntu is installed, there is no **root** password (page 598). To authenticate for sudo you must enter *your* password. When this book says you have to enter your password to authenticate (other than while logging in on a system), it assumes you are an administrator. If not, you must get an administrator to perform the task.

**adm** group The **adm** group can monitor the system but cannot run sudo. For example, a member of this group can display the **dmesg**, **kern** (kernel), and **syslog** (system log) files in **/var/log** whereas an ordinary user cannot. The name **adm** comes from the fact that the **/var/log** directory was originally named **/usr/adm** and subsequently **/var/adm**.

**admin** group Although it is mentioned in the **/etc/sudoers** file (page 607) for backward compatibility, as Ubuntu is installed there is no **admin** group (there is no entry for **admin** in **/etc/group** [page 484]).

**lpadmin** group Being a member of the **lpadmin** group allows you to add, modify, and delete printers but does not allow you to run sudo (page 542). The first user you set up is automatically made a member of the **lpadmin** group

# LE THE SPECIAL POWERS OF A USER WORKING WITH root PRIVILEGES

A user running with **root** privileges has the following powers—and more.

- Some commands, such as those that add new users, partition hard drives, and change system configuration, can be executed only by a user working with **root** privileges. Such a user can configure tools, such as sudo, to give

other specific users permission to perform tasks that are normally reserved for a user running with **root** privileges.

- Read, write, and execute file access and directory access permissions do not affect a user with **root** privileges. A user with **root** privileges can read from, write to, and execute all files, as well as examine and work in all directories.

  Exceptions to these privileges exist. For example, even a user working with **root** privileges cannot make sense of an encrypted file without possessing the key.

- Some restrictions and safeguards that are built into some commands do not apply to a user with **root** privileges. For example, a user with **root** privileges can change any user's password without knowing the old password.

**#** prompt

When you are running with **root** privileges in a command-line environment, by convention the shell displays a special prompt to remind you of your status. By default, this prompt is (or ends with) a hashmark (**#**).

### Console security

**security**

Linux is not secure from a person who has physical access to the computer. Additional security measures, such as setting boot loader and BIOS passwords, can help secure the computer. However, if someone has physical access to the hardware, as system console users typically do, it is very difficult to secure a system from that user.

### Least privilege

**caution**

When you are working with **root** privileges, perform any task while using the least privilege possible. When you can perform a task logged in as an ordinary user, do so. When you must run a command with **root** privileges, do as much as you can as an ordinary user, use su or sudo to give yourself **root** privileges, complete the part of the task that has to be done with **root** privileges, and revert to being an ordinary user as soon as you can. Because you are more likely to make a mistake when you are rushing, this concept becomes even more important when you have less time to apply it.



## LE GAINING root PRIVILEGES

Classically a user gained **root** privileges by logging in as the user named **root** or by giving an su (substitute user) command and providing the **root** password. More recently the use of sudo has taken over this classic technique of gaining **root** privileges. With sudo, the user logs in as herself, gives an sudo command, and provides her own password (not the **root** password) to gain **root** privileges. "Advantages of sudo" on page 603 discusses some of the advantages of sudo. By default, Ubuntu adds the ordinary user who was established when the system was installed (the first user) to the **sudo** group so she can run commands with **root** privileges.

Graphical environment

When an ordinary user executes a privileged command in a graphical environment, the system prompts for the **root** password or the user's password, depending on how the system is set up.

Ubuntu locks the **root** account by not assigning a password to the **root** account. As Ubuntu is installed, you cannot gain **root** privileges using a technique that requires you to supply the **root** password because no **root** password exists.

### There is a root account, but no root password

security As installed, Ubuntu locks the **root** account by not providing a **root** password. This setup prevents anyone from logging in to the **root** account (except when you bring the system up in recovery mode [page 450]). There is, however, a **root** account (a user with the username **root**—look at the first line in **/etc/passwd**). This account/user owns files (give the command **ls –l /usr/bin**) and runs processes (give the command **ps –ef** and look at the left column of the output). The **root** account is critical to the functioning of a Linux system.

When properly set up, the sudo utility enables you to run a command as though it had been run by a user logged in as **root**. This book uses the phrase **working with root privileges** to emphasize that, although you might not be logged in as **root**, when you use su or sudo you have the powers of the **root** user.

The following list describes some of the ways you can gain or grant **root** privileges. Some of these techniques depend on you supplying the password for the **root** account. Again, if the **root** account is locked, you cannot use these techniques. Other techniques depend on the **sudoers** file being set up to allow you to gain **root** privileges (page 607). If this file is not set up in this manner, you cannot use these techniques. As installed, the user who is set up when Ubuntu is installed (the first user) is a member of the **sudo** group and so can gain **root** privileges using sudo.

- You can give an su (substitute user) command while you are logged in as yourself. When you then provide the **root** password, you will be running with **root** privileges. See page 600.
- The sudo utility allows specified users to run selected commands with **root** privileges while they are logged in as themselves. You can set up sudo to allow certain users to perform specific tasks that require **root** privileges without granting these users systemwide **root** privileges. See page 602.
- You can log in as **root**. When you then supply the **root** password, you will be running with **root** privileges.
- When you bring the system up in recovery mode (page 450) you are logged in as the user named **root**.
- Some programs ask for a password (either your password or the **root** password) when they start or when you ask them to perform certain tasks. When you provide a password, the program runs with **root** privileges. The program stops running with root privileges after a certain amount of time or when you quit using the program. This setup keeps you from remaining logged in with **root** privileges when you do not need or intend to be.

LE+ Setuid file
- Any user can create a *setuid* (set user ID) file. Setuid programs run on behalf of the owner of the file and have all the access privileges the owner has.

While you are working with **root** privileges, you can change the permissions of a file owned by **root** to setuid. When an ordinary user executes a file that is owned by **root** and has setuid permissions, the program has *effective **root** privileges*. In other words, the program can do anything a program running with **root** privileges can do that the program normally does. The user's privileges do not change. When the program finishes running, all user privileges are as they were before the program started. Setuid programs owned by **root** are both extremely powerful and extremely dangerous to system security, which is why a system contains very few of them. Examples of setuid programs that are owned by **root** include passwd, at, and crontab. For more information refer to "Setuid and Setgid Permissions" on page 204 and "Real UID Versus Effective UID" (next).

### root-owned setuid programs are extremely dangerous

security

Because **root**-owned setuid programs allow someone who does not know the **root** password and cannot use sudo to gain **root** privileges, they are tempting targets for a malicious user. Also, programming errors that make normal programs crash can become **root** exploits in setuid programs. A system should have as few of these programs as possible. You can disable setuid programs at the filesystem level by mounting a filesystem with the **nosuid** option (page 500). See page 614 for a find command that lists all setuid files on the local system.

Logging in

The **/etc/securetty** file controls which terminals (ttys) a user can log in on as **root.** Using the **/etc/security/access.conf** file, PAM controls the who, when, and how of logging in. Initially this file contains only comments. See page 621, the comments in the file, and the **access.conf** man page for details.

## LE REAL UID VERSUS EFFECTIVE UID

UID and username

A UID (user ID) is the number the system uses to identify a user account. The **/etc/passwd** file associates a username with each UID. The username **root** is typically associated with UID 0. Most utilities display the associated username in place of a UID.

The kernel associates two UIDs with each process: a *real UID* and an *effective UID*. The third column of the **/etc/passwd** file (or NIS/LDAP) specifies your real UID. When you log in, your real UID is associated with the process running the login shell. Because you have done nothing to change it, the effective UID associated with the process running the login shell is the same as the real UID associated with that process.

Process privilege

The kernel determines which privileges a process has based on that process' effective UID. For example, when a process asks to open a file or execute a program, the kernel looks at the effective UID of the process to determine whether it is allowed to do so. When a user runs a setuid program (page 204), the program runs with the effective UID of the owner of the program, not the real UID of the user running the program.

Terminology: **root** privileges

When this book uses the phrase **run with root privileges** or **gain root privileges,** it means run with an effective UID of 0 (**root**).

# LE+ USING su TO GAIN root PRIVILEGES

### By default, under Ubuntu you cannot use su to gain root privileges

security During Ubuntu installation, no password is assigned to the **root** account (the account is locked) so you cannot use su to gain **root** privileges. There is no need for a **root** password; you can gain **root** privileges using sudo. See page 602 for more information on using sudo. If you want to use su to gain **root** privileges, you must first unlock (assign a password to) the **root** account as explained on page 613.

The su (substitute user) utility can spawn a shell or execute a program with the identity and privileges (effective UID) of a specified user, including **root**:

- Follow **su** on the command line with the name of a user; if you are working with **root** privileges or if you know the user's password, the newly spawned shell will take on the identity (effective UID) of that user.
- When you give an su command without an argument, su defaults to spawning a shell with **root** privileges (you have to supply the **root** password). That shell runs with an effective UID of 0 (**root**).

## SPAWNING A root SHELL

When you give an su command to work with **root** privileges, su spawns a new shell, which displays the **#** prompt. That shell runs with an effective UID of 0 (**root**). You can return to your normal status (and your former shell and prompt) by terminating the shell: Press CONTROL-D or give an **exit** command.

LE who, whoami The who utility, when called with the arguments **am i**, displays the real UID (translated to a username) of the process that called it. The whoami utility displays the effective UID (translated to a username) of the process that called it. As the following example shows, the su utility (the same is true for sudo) changes the effective UID of the process it spawns but leaves the real UID unchanged. (As Ubuntu is installed, the following examples will not work because there is no **root** password.)

```
$ who am i
sam      pts/0        2014-08-07 15:35 (192.168.206.1)
$ whoami
sam
$ su
Password:
# who am i
sam      pts/0        2014-08-07 15:35 (192.168.206.1)
# whoami
root
# exit
exit
$
```

LE id Giving an su command without any arguments changes your effective user and group IDs but makes minimal changes to the environment. For example, **PATH** has the same value as it did before you gave the su command. The id utility displays the effective UID and GID of the process that called it as well as the groups the process is associated with:

```
$ pwd
/home/sam
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
$ id
uid=1000(sam) gid=1400(pubs)
groups=1400(pubs),4(adm),24(cdrom),27(sudo),46(plugdev),108(lpadmin),124(sambashare)
$ su
Password:
# pwd
/home/sam
# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
# id
uid=0(root) gid=0(root) groups=0(root)
# exit
exit
$
```

When you give the command **su –** (you can use **–l** or **––login** in place of the hyphen), su provides a **root** login shell. It is as though you logged in as **root**. Not only do the shell's effective user and group IDs match those of **root**, but the environment is the same as when you log in as **root**. The login shell executes the appropriate startup files (page 335) before displaying a prompt, and the working directory is set to what it would be if you had logged in as **root** (**/root**). **PATH** is also set as though you had logged in as **root**. However, as who shows, the real UID of the new process is not changed from that of the parent process.

```
$ su -
Password:
# pwd
/root
# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
# who am i
sam      pts/0        2014-08-07 15:35 (192.168.206.1)
```

## EXECUTING A SINGLE COMMAND

You can use su with the **–c** option to run a command with **root** privileges, returning to the original shell when the command finishes executing. In the following example, Sam tries to display the **/etc/shadow** file while working as himself, a nonprivileged user. The cat utility displays an error message. When he uses su to run cat to display the file, su

prompts him for a password, he responds with the **root** password, and the command succeeds. The quotation marks are necessary because **su –c** takes the command it is to execute as a single argument.

```
$ cat /etc/shadow
cat: /etc/shadow: Permission denied

$ su -c 'cat /etc/shadow'
Password:
root:$6$lLQiMDmT$ES7L4ZEmxRxf.sO ... 5gSByz91:16289:0:99999:7:::
daemon:*:16177:0:99999:7:::
bin:*:16177:0:99999:7:::
sys:*:16177:0:99999:7:::
...
```

The next example first shows that Sam is not permitted to kill (page 455) a process. With the use of **su –c** and the **root** password, however, Sam is working with **root** privileges and is permitted to kill the process.

```
$ kill -15 4982
-bash: kill: (4982) - Operation not permitted
$ su -c "kill -15 4982"
Password:
$
```

The final example combines the **–** and **–c** options to show how to run a single command with **root** privileges in the **root** environment.

```
$ su -c pwd
Password:
/home/sam
$ su - -c pwd
Password:
/root
```



### Root privileges, PATH, and security

security The fewer directories you keep in **PATH** when you are working with **root** privileges, the less likely you will be to execute an untrusted program while working with **root** privileges. *Never include the working directory in **PATH** (as . or : : anywhere in **PATH**, or : as the last element of **PATH**).* For more information refer to "PATH: Where the Shell Looks for Programs" on page 365.

## LE+ USING sudo TO GAIN root PRIVILEGES

Ubuntu strongly encourages the use of sudo. In fact, as shipped, Ubuntu locks the **root** account (there is no password) so you cannot use su without adding a password for the **root** account. See page 613 for instructions on adding a password to the **root** account.

If sudo (www.sudo.ws) is not set up so you can use it, and a **root** password exists and you know what it is, see "Administrator and the **sudo** group" on page 604 for instructions on setting up sudo so you can use it to gain **root** privileges.

## ADVANTAGES OF sudo

As the following list explains, using sudo rather than the **root** account for system administration offers many advantages.

- When you run sudo, it requests *your* password—not the **root** password—so you have to remember only one password.

### You can use sudo to perform any task you can perform working as root

security Many administrators unlock the **root** account without a good reason. Some do so because of a lack of familiarity with sudo. Others do it after giving an incorrect sudo command and receiving an error; the administrator concludes that sudo is to blame. If you are not familiar with sudo, try learning about it and using it for a while. **Use sudo to perform any task you can perform using su.**

- The sudo utility logs all commands it executes. This log can be useful for retracing your steps for system auditing or if you make a mistake.
- The sudo utility logs the username of a user who issues an sudo command. On systems with more than one administrator, this log tells you which users have issued sudo commands. Without sudo, you would not know which user issued a command while working with **root** privileges.
- The sudo utility allows implementation of a finer-grained security policy than does the use of su and the **root** account. Using sudo, you can enable specific users to execute specific commands—something you cannot do with the classic **root** account setup.
- Using sudo makes it harder for a malicious user to gain access to a system. When there is an unlocked **root** account, a malicious user knows the username of the account she wants to crack before she starts. When the **root** account is locked, the user has to determine a username *and* the password to break into a system.
- Managing **root** passwords over many systems is challenging. Remembering each system's password is difficult without writing them down (and storing them in a safe place), and then retrieving a password is time consuming. Keeping track of which users know which **root** passwords is impossible. Using sudo, even for full **root**-shell access, makes the tasks of gaining **root** privileges on a large number of systems and tracking who can gain **root** privileges on each system much easier.



## SECURITY OF sudo

Some users question whether sudo is less secure than su. Because both rely on passwords, they share the same strengths and weaknesses. If the password is compromised, the system is compromised. However, if the password of a user who is allowed by sudo to do one task is compromised, the entire system might not be at risk. Thus, *if used properly,* the finer granularity of sudo's permissions structure *can* make it a more secure tool than su. Also, when sudo is used to invoke a single command, it is less likely that a user will be tempted to keep working with **root** privileges than if the user opens a **root** shell using su.

Using sudo might not always be the best, most secure way to set up a system. On a system used by a single user, there is not much difference between using sudo and carefully using su and a **root** password. In contrast, on a system with several users, and especially on a network of systems with central administration, sudo can be set up to be more secure than su.

### Use pkexec to run graphical programs

**caution** Use pkexec (a graphical front end for PolicyKit) when you run a graphical program that requires **root** privileges. Alternately, you can use sudo with or without the **–i** option. Just as sudo does, pkexec requests *your* password. The use of gksudo (**gksu** package) is discouraged; this utility might be removed from future releases of Ubuntu.

## USING sudo

When you install Ubuntu, the first user you set up is included in the **sudo** group. As installed, sudo is configured to allow members of the **sudo** group to use sudo to run programs with **root** privileges. Because there is no **root** password, initially the only way to perform privileged administrative tasks from the command line is for the first user to run them using sudo. Graphical programs call other programs, such as pkexec (see the adjacent tip), which in turn call sudo for authentication.

Administrator and the **sudo** group

When you add or modify a user, you can specify that the user is an administrator by making the user a member of the **sudo** group. When you specify an account type of Administrator in the User Accounts window (Figure 4-16, page 118), the user becomes a member of the **sudo** group.

Adding a user to the **sudo** group

While working with **root** privileges, run usermod with the **–a** and **–G sudo** options to add the user to the **sudo** group. Substitute a username for **zach** in the following example.

```
$ sudo usermod -a -G sudo zach
$ grep sudo /etc/group
sudo:x:27:sam,max,zach
```

The grep command shows Zach is a member of the **sudo** group. See **/etc/group** on page 484 for more information on groups.

Timestamp

By default, sudo asks for *your* password (not the **root** password) the first time you run it. At that time, sudo sets your *timestamp*. After you supply a password, sudo will not prompt you again for a password for 15 minutes (by default), based on your timestamp.

#### EXECUTING A SINGLE COMMAND

In the following example, Sam tries to set the system clock while working as himself, a nonprivileged user. The date utility displays an error message followed by the expanded version of the date Sam entered. When he uses sudo to run date to set the system clock, sudo prompts him for his password, and the command succeeds.

```
$ date 12121500
date: cannot set date: Operation not permitted
Tue Jun 24 19:28:00 PDT 2014

$ sudo date 06241928
Tue Jun 24 19:28:00 PDT 2014
```

Next Sam uses sudo to unmount a filesystem. Because he gives this command within 15 minutes of the previous sudo command, he does not need to supply a password:

```
$ sudo umount /music
$
```

Now Sam uses the **–l** option to check which commands sudo will allow him to run. Because the **sudoers** file is set up as explained in "Administrator and the **sudo** group," on the previous page, and because Sam was the first user registered on the system (and is therefore a member of the **sudo** group), he is allowed to run any command as any user or group.

```
$ sudo -l
...
User sam may run the following commands on guava:
    (ALL : ALL) ALL
```

### Gaining root privileges to edit a file

tip With the **–e** option, or when called as sudoedit, sudo edits with **root** privileges the file named by its argument. By default sudo uses the nano editor; see the **–e** option on page 607 for instructions on how to specify a different editor.

Any user who can use sudo to run commands with **root** privileges can use the **–e** option. To give other users permission to edit any file using **root** privileges, specify in the **sudoers** file that that user can execute the command **sudoedit**. For more information refer to "User Privilege Specifications" on page 608.

Calling an editor in this manner runs the editor in the user's environment, maintaining the concept of least privilege. The sudo utility first copies the file to be edited to a temporary file that is owned by the user. If the file does not exist, sudo creates a new file that is owned by the user. When the user has finished editing the file, sudo copies it back in place of the original file (maintaining the original permissions).



### SPAWNING A root SHELL

When you have several commands you need to run with **root** privileges, it might be easier to spawn a **root** shell, give the commands without having to type **sudo** in front of each one, and exit from the shell. This technique defeats some of the safeguards built into sudo, so use it carefully and remember to return to a non**root** shell as soon as possible. See the caution on least privilege on page 597. Use the sudo **–i** option (page 607) to spawn a **root** shell:

```
$ pwd
/home/sam
$ sudo -i
# id
uid=0(root) gid=0(root) groups=0(root)
# pwd
/root
# exit
logout
$
```

In this example, sudo spawns a **root** shell, which displays a # prompt to remind Sam that he is working with **root** privileges. The id utility displays the effective UID of the user running the shell. The exit command (you can also use CONTROL-D) terminates the **root** shell, returning Sam to his normal status and his former shell and prompt.

sudo's environment

The pwd builtin in the preceding example shows one aspect of the modified environment created by the **–i** option. This option spawns a **root** login shell (a shell with the same environment as a user logging in as **root** would have) and executes **root's** startup files (page 335). Before issuing the **sudo –i** command, the pwd builtin shows **/home/sam** as Sam's working directory; after the command, it shows **/root**, **root**'s home directory, as the working directory. Use the **–s** option (page 607) to spawn a **root** shell without modifying the environment. When you call sudo without an option, it runs the command you specify in an unmodified environment. To demonstrate this feature, the following example calls sudo without an option to run pwd. The working directory of a command run in this manner does not change.

```
$ pwd
/home/sam
$ sudo pwd
/home/sam
```

Redirecting output

The following command fails because, although the shell that sudo spawns executes ls with **root** privileges, the nonprivileged shell that the user is running redirects the output. The user's shell does not have permission to write to **/root**.

```
$ sudo ls > /root/ls.sam
-bash: /root/ls.sam: Permission denied
```

There are several ways around this problem. The easiest is to pass the whole command line to a shell running under sudo:

```
$ sudo bash -c "ls > /root/ls.sam"
```

The bash **–c** option spawns a shell that executes the string following the option and then terminates. The sudo utility runs the spawned shell with **root** privileges. You must quote the string to prevent the nonprivileged shell from interpreting special characters. You can also spawn a **root** shell using **sudo –i**, execute the command, and exit from the privileged shell. (See the preceding paragraphs.)

**optional**

Another way to deal with the issue of redirecting output of a command run by sudo is to use tee (page 170):

```
$ ls | sudo tee /root/ls.sam
...
```

This command line sends standard output of ls through sudo to tee, which copies its standard input to the file named by its argument and to standard output. If you do not want to display the output, you can have the nonprivileged shell redirect the output to **/dev/null** (page 481). The next example uses this technique to do away with the displayed output and uses the **–a** option to tee to append to the file instead of overwriting it:

```
$ ls | sudo tee -a /root/ls.sam > /dev/null
```

## OPTIONS

You can use command-line options to control how sudo runs a command. Following is the syntax of an sudo command line:

*sudo [**options**] [**command**]*

where ***options*** is one or more options and ***command*** is the command you want to execute. Without the **–u** option, sudo runs ***command*** with **root** privileges. Some of the more common ***options*** follow; see the sudo man page for a complete list.

**–b** (**background**) Runs ***command*** in the background.

**–e** (**edit**) Causes sudo to edit the file named ***command*** (***command*** is a filename and not a command) with **root** privileges using the editor named by the **SUDO_EDITOR**, **VISUAL**, or **EDITOR** environment variable. Default is the nano editor.

**–g** ***group*** Runs ***command*** with the privileges of ***group***.

**–i** (**initial login environment**) Spawns the shell that is specified for **root** (or another user specified by **–u**) in **/etc/passwd**, running **root**'s (or the other user's) startup files, with some exceptions (e.g., **TERM** is not changed). See "Spawning a **root** Shell" on page 605 for an example.

**–k** (**kill**) Resets the timestamp (page 604) of the user running the command, which means the user must enter her password the next time she runs sudo.

**–l** (**list commands**) Lists the commands the user who is running sudo is allowed to run on the local system. Does not take a ***command***.

**–s** (**shell**) Spawns a new **root** (or another user specified by **–u**) shell as specified in the **/etc/passwd** file. Similar to **–i** but does not change the environment. Does not take a ***command***.

**–u** ***user*** Runs ***command*** with the privileges of ***user***. Without this option, sudo runs ***command*** with **root** privileges.



## LPI sudoers: CONFIGURING sudo

As installed, sudo is not as secure and robust as it can be if you configure it carefully. The sudo configuration file is **/etc/sudoers**. You can edit this file to give specific users the ability to run only certain commands with **root** privileges as opposed to any commands. You can also limit these users to running commands only with certain options or arguments. Or you can set up sudo so a specific user cannot use a specific option or argument when running a command with **root** privileges.

The best way to edit **sudoers** is to use visudo by giving the command: **su –c visudo** or **sudo visudo**. The visudo utility locks, edits, and checks the grammar of the **sudoers** file. By default, visudo calls the nano editor. Add the following line to the **sudoers** file (or to one of the files in the **/etc/sudoers.d** directory) to cause visudo to use the vim editor (page 270):

```
Defaults editor="/usr/bin/vi"
```

Replace **/usr/bin/vi** with the pathname of the textual editor of your choice.

BLANK

# 21

# postfix: Setting Up Mail Servers, Clients, and More



## In This Chapter



## Objectives

After reading this chapter you should be able to:

- Explain what **postfix** is
- Explain the purpose and give examples of an MUA, an MTA, and an MDA
- Describe the role of SMTP, IMAP, and POP
- Configure the local SMTP server to send outgoing mail directly to destination servers
- Configure the local SMTP server to use Gmail as a smarthost
- Configure mail aliases and mail forwarding
- Install and configure SpamAssassin on a mail client and a mail server
- Install and configure a Webmail service (SquirrelMail)
- Set up a mailing list (Mailman)
- Install and configure an IMAP server (**dovecot**)

# OVERVIEW

MUA Sending and receiving email require three pieces of software. At each end, there is an email client (or reader), more formally called an MUA (mail user agent). A user uses an MUA to send, access, and manage her email. Common MUAs are Evolution, KMail, Thunderbird, mutt, and Outlook.

MTA, MDA When you send an email, the MUA hands it to an MTA (mail transfer agent, such as **postfix**, **exim4**, or **sendmail**), which directly or indirectly transfers it to the destination server. At the destination, an MDA (mail delivery agent, such as **postfix** or **procmail**) puts the mail in the recipient's mailbox. An MUA on the receiving system retrieves email from the mailbox and allows a user to read, organize, store, and delete email.

Alternately, a single, local program can function as both an MTA and an MDA. The **dovecot** mail server can retrieve email from a remote system, such as an ISP's mail server, using POP3 (Post Office Protocol) or IMAP (Internet Message Access Protocol) and can then put the email in the recipients' mailboxes. See the next page for more information on POP3 and IMAP.

Web-based MUA Lastly, Web-based MUAs such as Gmail and Yahoo! Mail serve the same function as locally installed MUAs. This type of email is referred to as Webmail.

SMTP Most Linux MUAs expect a local MTA such as **postfix** to deliver outgoing email. On some systems, including those with a dial-up connection to the Internet, the MTA sends email to an ISP's mail server. Because most MTAs use SMTP (Simple Mail Transfer Protocol) to deliver email, they are often referred to as SMTP servers. By default, when you install **postfix** on an Ubuntu system, **postfix** uses its builtin MDA to deliver email to the recipient's mailbox file.



### You do not need to set up postfix to send and receive email

tip Most MUAs can use POP or IMAP to receive email from an ISP's server. These protocols do not require an MTA such as **postfix**. As a consequence, you do not need to install or configure **postfix** (or another MTA) to receive email. Although you still need SMTP to send email, the SMTP server can be at a remote location, such as your ISP. Thus you might not need to concern yourself with setting up an SMTP server to send email either.

### If you want to run sendmail

tip Give the following command if you want to run **sendmail**.

```
$ sudo apt-get install sendmail
```

This command stops and removes **postfix** and then installs and starts **sendmail**.

# MAILBOXES: mbox VERSUS maildir FORMAT

Two common formats for email mailboxes are **mbox** and **maildir**. The **postfix** mail server defaults to using the **mbox** format. See **home_mailbox** on page 790 for information on changing this default.

The **mbox** format holds all messages for a user in a single file, typically in **/var/mail**. To prevent corruption, a process must lock this file while it is adding messages to or deleting messages from the file; thus the MUA cannot delete a message at the same time the MTA is adding messages. A competing format, **maildir**, holds each message in a separate file. This format does not use locks, allowing an MUA to delete messages from a user at the same time as mail is delivered to the same user. In addition, the **maildir** format is better able to handle larger mailboxes. The downside is that the **maildir** format adds overhead when you are using a protocol such as IMAP to check messages. Most MTAs support both **mbox** and **maildir** formats.

## PROTOCOLS: IMAP AND POP3

Two protocols are commonly used by MUAs to retrieve and store email: POP3 (Post Office Protocol) and IMAP (Internet Message Access Protocol).

POP3 downloads all email from the server and stores it locally, in the mailbox file or directory on the local system. The email is available whether or not the local system is connected to the Internet, but it is available only from the computer you downloaded it on.

IMAP syncs local mail storage with the server; email stays on the server (unless you remove it) and you can organize it into folders to make it easier to work with. Many email client programs initially download headers for your review and only download the message bodies when you ask to read an email. IMAP allows you to view email from different computers but does not give you access unless the local system is connected to the Internet.

Both protocols have more secure versions that use TLS encryption to connect to the mail server: POP3S and IMAPS.



## LPI INTRODUCTION TO postfix

When the network that was to evolve into the Internet was first set up, it connected a few computers, each serving a large number of users and running several services. Each computer was capable of sending and receiving email and had a unique hostname, which was used as a destination for email.

Today the Internet has a large number of transient clients. Because these clients do not have fixed IP addresses or hostnames, they cannot receive email directly. Users on these systems usually maintain an account on an email server run by their employer or an ISP, and they collect email from this account using POP or IMAP. Alternately, many people use Webmail services such as Gmail. Unless you own a domain where you want to receive email, you will not need to set up **postfix** to receive mail from nonlocal systems.

The **postfix** system (www.postfix.org) is an alternative MTA that is fast and easy to administer but is compatible enough with **sendmail** to not upset **sendmail** users. It has a good reputation for ease of use and security and is a drop-in replacement for **sendmail**.

# OUTBOUND EMAIL

When used as a server, **postfix** accepts outbound email and directs it to the system it is addressed to (the destination system). This section describes the different ways you can set up **postfix** to perform this task.

## ACCEPTING EMAIL FOR DELIVERY

You can set up a **postfix** server so it accepts outbound email from the local system only, from specified systems (such as a LAN), or from all systems. Accepting email from unknown systems makes it likely the server will propagate spam.

## DELIVERING EMAIL

Direct connection

As installed, **postfix** connects directly to SMTP servers on nonlocal destination systems. This SMTP server then delivers the email. Typically, **postfix** delivers local email directly.

Smarthost

You can configure **postfix** so it that sends email bound for nonlocal systems to an SMTP server that relays the mail to its destination. This type of server is called a *smarthost* or *SMTP relay*.

Port 25

By default, SMTP uses port 25. Some Windows viruses use a simple, self-contained SMTP server to propagate themselves. As a partial defense against these types of viruses, some ISPs and larger organizations block all outgoing connections that originate or terminate on port 25, with the exception of connections to their own SMTP server (smarthost). Blocking this port prevents local systems from making a direct connection to send email. These organizations require you to use a smarthost for email bound for nonlocal systems.



# INBOUND EMAIL

You can configure **postfix** to accept email for a registered domain name as specified in the domain's DNS MX record (page 899). However, most mail clients (MUAs) do not interact directly with **postfix** to receive email. Instead, they use POP or IMAP—protocols that include features for managing mail folders, leaving messages on the server, and reading only the subject of an email without downloading the entire message. If you want to collect email from a system other than the one running the incoming mail server, you might need to set up a POP or IMAP server, as discussed on page 807.

# LPI THE postfix TO sendmail COMPATIBILITY INTERFACE

The **postfix** to **sendmail** compatibility interface translates a few commands from **sendmail** format so they can be used with **postfix**. For example, the classic **sendmail** newaliases utility works with **postfix** because it calls the program named sendmail, which is not the traditional **sendmail**, but rather the **postfix** to **sendmail** compatibility interface.

```
$ ls -l $(which newaliases)
lrwxrwxrwx 1 root root 16 Feb 11 22:23 /usr/bin/newaliases -> ../sbin/sendmail

$ man sendmail
SENDMAIL(1)              General Commands Manual              SENDMAIL(1)

NAME
        sendmail - Postfix to Sendmail compatibility interface
...
```

The **postfix** to **sendmail** compatibility interface also includes an implementation of the mailq utility (page 786). You can also use the sendmail utility to test **postfix** (page 785).

## LPI ALTERNATIVES TO postfix

**LPI sendmail** The most popular MTA today, **sendmail** (**sendmail** package; www.sendmail.org) first appeared in 4.1BSD.

Over the years, **sendmail** has grown to be enormously complex. Its complexity makes it challenging to configure if you want to set up anything more than a simple mail server. Although its complexity allows **sendmail** to be flexible and to scale well, its size and complexity also add to its vulnerability.

Unlike **sendmail**, which uses a single binary to handle all aspects of inbound and outbound mail, **postfix** is more modular. This setup allows **postfix** to run select, smaller binaries as needed. Not only is this configuration more secure, but it is more resilient: An exploit can affect only a module, not the entire server.

**LPI exim4** The **exim4** (www.exim.org; **exim** package) mail server was introduced in 1995 by the University of Cambridge. It can be configured in several different ways and includes many features other MTAs lack.

**LPI Qmail** Qmail (www.qmail.org) is a direct competitor of **postfix** and has the same objectives. By default, Qmail stores email using the **maildir** format (sometimes referred to as "Qmail fomat," as opposed to the **mbox** format that other MTAs use (page 780).



## MORE INFORMATION

Web **postfix**: postfix.org/documentation.html
**sendmail**: sendmail.com
**exim4**: exim.org
**procmail**: procmail.org
IMAP and POP3: dovecot.org, cyrusimap.org
STARTTLS: www.fastmail.fm/help/technology_ssl_vs_tls_starttls.html, starttls.info
SpamAssassin: spamassassin.apache.org, wiki.apache.org/spamassassin
Bayesian classifier: wiki.apache.org/spamassassin/BayesInSpamAssassin
Mailman: list.org
SquirrelMail: squirrelmail.org
**dovecot**: dovecot.org, wiki.dovecot.org, wiki2.dovecot.org/TestInstallation, wiki2.dovecot.org/SSL/DovecotConfiguration
Qmail: qmail.org

Local man pages: **postfix**, postconf(5), postmap, local, sendmail, postalias, **master**, **master**(5) spamassassin, spamc, **spamd**
**postfix**: **/usr/share/postfix/main.cf.dist** (a sample, well-commented copy of the configuration file)
**dovecot**: **/usr/share/dovecot**, **/usr/share/doc/dovecot***
**mailman**: **/usr/share/doc/mailman**
**squirrelmail**: **/usr/share/doc/squirrelmail**
SpamAssassin: **/usr/share/doc/spamassassin***, after installing the **perl-doc** package, give the following command:

```
$ perldoc Mail::SpamAssassin::Conf
```

## LPI SETTING UP A postfix MAIL SERVER

This section explains how to set up a **postfix** mail server.

### PREREQUISITES

Install the following packages:

- **postfix**
- **mailutils** (optional, includes programs for handling email)

When you install the **postfix** package, the **dpkg postinst** script displays two pseudo-graphical windows. The first asks you to specify the type of mail configuration you want to set up. For the purposes of this section, leave the highlight on the default **Internet Site** and press RETURN. The second asks you to specify the domain name for the local (server) system; this value goes in the **/etc/mailname** file. The script suggests the *FQDN* (page 1248; e.g., guava.example.com) of the system. Make sure that this value is a valid address; it is the value that will appear on all outbound email that originates on the local system. Some systems bounce email that comes from an unknown address, others mark the email as spam.

**postfix** init script As installed, **postfix** is active and enabled so it will run when the system is booted. After changing its configuration files, give the following command to reload **postfix**, causing it to reread its configuration files:

```
$ sudo service postfix reload
```

You can specify **stop** in place of **reload** to **stop postfix** in an orderly manner, or **status** to display the status of **postfix**. Use **reload** to refresh **postfix** after changing its configuration, not **stop** and **start**.

### NOTES

Firewall An SMTP server normally uses TCP ports 25 and 587. SMTP connections secured by SSL (SMTPS) use port 465. If an SMTP server system that receives nonlocal mail

is running a firewall, you need to open one or more of these ports. Give the following commands to open these ports; see page 924 for information on ufw.

```
$ sudo ufw allow 25/tcp
$ sudo ufw allow 587/tcp
$ sudo ufw allow 465
```

Log files You must be a member of the **adm** group or work with **root** privileges to display the **/var/log/mail.log** and **/var/log/mail.err** files.

# TESTING postfix

This section explains how to test a **postfix** setup.

## LPI OUTBOUND EMAIL

As long as **postfix** can connect to port 25 outbound, you should not need to further configure **postfix** after you install it. You can use the sendmail utility, which is part of the **postfix** to **sendmail** compatibility interface (page 782), to test **postfix**. (If you have installed the **mailutils** package, you can use mail.) With **postfix** running, give the following command, substituting the email address of someone on a remote system for **sam@example.com**.

```
$ echo 'testing postfix' | sendmail sam@example.com
```

In addition to checking to see whether the person received the email, you can display the end of the **/var/log/mail.log** file to see whether the command worked. See the next page for more information on **postfix** log files.

## INBOUND EMAIL

You can use telnet to test the **postfix** inbound mail setup. The following example uses telnet to connect to the SMTP port (port 25) on the local system (localhost). You can perform the same test from a remote system; make sure the firewall on the server allows the appropriate packets in (previous page).

In the following example, Sam gives a **telnet** command, specifying the system to connect to as **localhost** and the port as **smtp** (he could have specified 25). The SMTP server (**postfix**) responds and waits for input. Because **postfix** expects to be talking to a machine, it expects your replies to follow a very specific order and syntax. If you use the wrong syntax, **postfix** displays an error as shown. Because the order is always the same, it does not prompt for input but waits silently.

The first thing **postfix** wants to see is the command **ehlo** followed by the FQDN of the system that is connecting to the **postfix** server. After receiving the error message, Sam gives an **ehlo** command, specifying the system as **machine.domain.com**. The **postfix** server responds by displaying some information about itself and waits for more input.

Sam then tells **postfix** where the mail is from and where it is going; **postfix** acknowledges each input string with **Ok**. The command **data** tells **postfix** that the body of the email

will follow; **postfix** responds by telling you to end the email with a line that has nothing but a period on it. Sam types his message, a line with a period, and then the command **quit**. After telnet closes the connection, the shell tells Sam that he has mail—the mail he just sent himself. He gives the shell a **mail** command (part of the **mailutils** package), and mail displays the header from the mail he just sent himself.

```
$ telnet localhost smtp
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 guava.example.com ESMTP Postfix (Ubuntu)
my name is Sam
502 5.5.2 Error: command not recognized
ehlo machine.domain.com
250-guava.example.com
...
250 DSN
mail from:<max@machine.domain.com>
250 2.1.0 Ok
rcpt to:<sam@localhost>
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
Here is a note to myself.
.
250 2.0.0 Ok: queued as C337113F669
quit
221 2.0.0 Bye
Connection closed by foreign host.
You have new mail in /var/mail/sam

$ mail
"/var/mail/sam": 1 message 1 new
>N   1 max@machine.domain Thu Aug 14 16:06  11/430
...
```

See page 310 for another example of using telnet to connect to an SMTP port.

LPI mailq The mailq utility (part of the **postfix** to **sendmail** compatibility interface [page 782]) displays the status of the outgoing mail queue. When there are no messages in the queue, it reports the queue is empty. Unless they are transient, messages in the queue usually indicate a problem with the local or remote MTA configuration or a network problem.

```
# mailq
Mail queue is empty
```

## postfix LOG FILES

**/var/log/mail.log** The **postfix** daemon stores log messages in **/var/log/mail.log** and error messages in **/var/log/mail.err**. Following is a sample log entry:

```
$ cat /var/log/mail.log
guava postfix/master[3672]: daemon started -- version 2.11.0, configuration /etc/postfix
guava postfix/pickup[3675]: 4575719FD63: uid=1000 from=<sam>
guava postfix/cleanup[3836]: 4575719FD63: message-id=<20140402230826.4575719FD63@guava.example.com>
guava postfix/qmgr[3676]: 4575719FD63: from=<sam@example.com>, size=279, nrcpt=1 (queue active)
guava postfix/smtp[3838]: 4575719FD63: to=<mgs@sobell.com>, relay=sobell.com[216.38.53.176]:25,
delay=1.6, delays=0.02/0.01/0.46/1.1, dsn=2.0.0, status=sent (250 OK id=1WVUHM-0004La-Nq)
guava postfix/qmgr[3676]: 4575719FD63: removed
```

Each line of a log entry starts with a timestamp (removed from the example), the name of the system sending the email (**guava**), the name of the mail server (**postfix**), the phase of the process the line is reporting on (e.g., **pickup**, **qmgr** (queue manager), **smtp**), the PID of that phase, and the queue ID (**4575719FD63**). The address of the sender follows **from=** label in the **qmgr** line. The address of the recipient follows the **to=** label in the long, wrapped **smtp** line. Additional fields on this line provide the name of the mailer and the time it took to send the message. If a message is sent correctly, the **stat=** label is followed by **sent**. After successfully sending a message, the **postfix** queue manager removes it from the queue.

If you send and receive a lot of email, the mail log files can grow quite large. The logrotate (page 582) **rsyslog** entry archives and rotates these files regularly.

## LPI JUMPSTART: CONFIGURING postfix TO USE GMAIL AS A SMARTHOST



### You might not need to configure postfix

tip If the test at "Testing postfix" on page 785 works immediately after you install **postfix**, you do not need to further configure **postfix**.

This JumpStart configures **postfix** to use your Gmail account to relay outbound email to its destination. In place of Gmail you can use an ISP or an outgoing SMTP email service (e.g., www.authsmtp.com). This **postfix** server:

- Uses the Gmail SMTP server to relay outbound email to its destination (a smarthost or SMTP relay).
- Sends to the SMTP server email originating from the local system only. It does not forward email originating from other systems. (See **mynetworks** on page 790 if you want to forward email from other systems.)
- Handles inbound email for the local system.

To set up this server, you must edit the **/etc/mailname** and **/etc/postfix/main.cf** files, create the **tls_policy.db** and **sasl_passwd.db** hash files, and reload **postfix** (page 784). Make sure you followed the instructions under "Prerequisites" on page 784 while installing **postfix**.

1. Edit **/etc/mailname**—Create or edit the **/etc/mailname** file so that it holds the FQDN that **postfix** will assign to outbound email. For example:

```
$ cat /etc/mailname
guava.example.com
```

2. Edit **/etc/postfix/main.cf**—Initially, the **relayhost** configuration parameter is assigned the null value (nothing follows the equal sign). Working with

BLANK

# D

# LPI and CompTIA Certification



### In This Appendix



This book is used as the adopted text in many college classes. Because students who take these classes often seek LPI or CompTIA certification, instructors have asked for a mapping of certification objectives to the material covered in this book. This book fully covers LPI's Linux Essentials certification learning goals and provides extensive coverage of CompTIA's Linux+ exam objectives. This appendix maps these learning goals and exam objectives to pages in this book. The following icons are used throughout the book to mark the places where learning goals and exam objectives are discussed.

***LPI*** This icon indicates coverage of a topic in the CompTIA's Linux+ exam objectives.

***LE*** This icon indicates coverage of a topic in the LPI's Linux Essentials certification learning goals.

***LE+*** This icon indicates coverage of a topic in the CompTIA's Linux+ exam objectives and a topic in the LPI's Linux Essentials certification learning goals.

# MORE INFORMATION

LPI Linux Essentials: www.lpi.org/linux-certifications/introductory-programs/linux-essentials
LPI Certification Exams: www.lpi.org/linux-certifications/programs/lpic-1
CompTIA Exams: certification.comptia.org/getCertified/certifications/linux.aspx
CompTIA and LPI partnership: www.lpi.org/linux-certifications/partnership-programs/comptia

# LINUX ESSENTIALS

## TOPIC 1: THE LINUX COMMUNITY AND A CAREER IN OPEN SOURCE

### 1.1 LINUX EVOLUTION AND POPULAR OPERATING SYSTEMS

#### DESCRIPTION: KNOWLEDGE OF LINUX DEVELOPMENT AND MAJOR DISTRIBUTIONS

**Key Knowledge Areas**
Open Source Philosophy
- Open-Source Software and Licensing page 6

Distributions
- Distribution page 6

Embedded Systems
- Embedded and mobile Linux page 6

**Partial List of Used Files, Terms, and Utilities**
Android
- Embedded and mobile Linux page 6

Debian
- Distribution page 6

CentOS



### 1.2 MAJOR OPEN SOURCE APPLICATIONS

#### DESCRIPTION: AWARENESS OF MAJOR APPLICATIONS AND THEIR USES

**Key Knowledge Areas**
Desktop Applications
- Desktop Applications page 1173

Server Applications
- page 464
- Chapter 13: Printing with CUPS page 539
- Chapter 18: The OpenSSH Secure Communication Utilities page 713
- Chapter 20: FTP: Transferring Files Across a Network page 753
- Chapter 21: **postfix**: Setting Up Mail Servers, Clients, and More page 779
- Chapter 22: NIS and LDAP page 813
- Chapter 23: NFS: Sharing Directory Hierarchies page 843
- Chapter 24: Samba: Linux and Windows File and Printer Sharing page 869

- Chapter 25: DNS/BIND: Tracking Domain Names and Addresses page 891
- Chapter 27: Apache (apache2): Setting Up a Web Server page 951

Mobile Applications
- Embedded and mobile Linux page 6

Development Languages
- Chapter 28: Programming the Bourne Again Shell (bash) page 1003
- Chapter 29: The Python Programming Language page 1103
- Chapter 30: The MariaDB SQL Database Management System page 1135

Package Management Tools and repositories
- Chapter 12: Finding, Downloading, and Installing Software page 509
- Appendix C: Keeping the System Up to Date Using yum page 1177

**Partial List of Used Files, Terms, and Utilities**

OpenOffice.org, LibreOffice, Thunderbird, Firefox, Blender, Gimp, Audacity, ImageMagick
- Desktop Applications page 1173

Apache, MySQL, PostgreSQL
- Chapter 27: Apache (apache2): Setting Up a Web Server page 951
- Chapter 30: The MariaDB SQL Database Management System page 1135
- Programming languages page 1174

NFS, Samba, OpenLDAP, **postfix**, DNS, DHCP
- Chapter 21: **postfix**: Setting Up Mail Servers, Clients, and More page 779
- Chapter 23: NFS: Sharing Directory Hierarchies page 843
- Chapter 24: Samba: Linux and Windows File and Printer Sharing page 869
- Chapter 25: DNS/BIND: Tracking Domain Names and Addresses page 891
- DHCP: Configures Network Interfaces page 464
- Introduction to LDAP page 830

C, Perl, shell, Python, PHP
- Chapter 28: Programming the Bourne Again Shell (bash) page 1003
- Chapter 29: The Python Programming Language page 1103
- Programming Languages page 1174



## 1.3 UNDERSTANDING OPEN SOURCE SOFTWARE AND LICENSING

### DESCRIPTION: OPEN COMMUNITIES AND LICENSING OPEN SOURCE SOFTWARE FOR BUSINESS

**Key Knowledge Areas**

Licensing
- Open-Source Software and Licensing page 6

Free Software Foundation (FSF), Open Source Initiative (OSI)
- GNU Project page 3
- Linux Is More Than a Kernel page 6
- FOSS/FLOSS page 7
- GNOME and KDE page 17

**Partial List of Used Files, Terms, and Utilities**

GPL, BSD, Creative Commons
- Berkeley UNIX (BSD) page 3

- GPL page 5
- Creative Commons page 1242 (Glossary)

Free Software, Open Source Software, FOSS, FLOSS

- FOSS/FLOSS page 7

Open Source business models

- Making money page 7

## 1.4 ICT SKILLS AND WORKING IN LINUX

### DESCRIPTION: BASIC INFORMATION AND COMMUNICATION TECHNOLOGY (ICT) SKILLS AND WORKING IN LINUX

**Key Knowledge Areas**

Desktop Skills

- Chapter 4: Introduction to Ubuntu page 97

Getting to the Command Line

- Working from the Command Line page 125
- Chapter 7: The Linux Utilities page 223

Industry uses of Linux, Cloud Computing, and Virtualization

- Chapter 17: Setting Up Virtual Machines Locally and in the Cloud page 687

**Partial List of Used Files, Terms, and Utilities**

Using a browser, privacy concerns, configuration options, searching the Web, and saving content

- Firefox: support.mozilla.org/en-US/products/firefox
- Chrome: www.google.com/intl/en/chrome/browser/features.html
- Opera: www.opera.com

Terminal and Console

- Using a Virtual Console page 127

Password issues

- User Accounts: Changing Your Account Type and Password (GUI) page 118
- Password Security page 143
- `passwd`: Changing Your Password (CLI) page 144
- Passwords page 615

Privacy issues and tools

- Search the Web for **browser privacy**
- Mozilla: support.mozilla.org/en-US/kb/private-browsing-browse-web-without-saving-info
- lifehacker.com/the-best-browser-extensions-that-protect-your-privacy-479408034

Use of common open-source applications in presentations and projects

- Desktop Applications page 1173

# TOPIC 2: FINDING YOUR WAY ON A LINUX SYSTEM

## 2.1 COMMAND LINE BASICS

### DESCRIPTION: BASICS OF USING THE LINUX COMMAND LINE

**Key Knowledge Areas**

Basic shell
- Working from the Command Line page 125
- Chapter 5: The Shell page 149

Formatting commands
- The Command Line page 152

Working with Options
- Options page 153

Variables
- Parameters and Variables page 358
- Variables page 1053

Globbing
- Filename Generation/Pathname Expansion page 173
- Pathname Expansion page 418

Quoting
- Special Characters page 150
- Quoting the $ page 360
- Quotation marks page 418

**Partial List of Used Files, Terms, and Utilities**

echo
- echo: Displays Arguments page 227
- **echo –e** page 1031

history
- History page 382

**PATH** env variable
- Set **PATH** in **.bash_profile** page 337
- **PATH**: Where the Shell Looks for Programs page 365

which
- which page 263

**Nice to Know**

Substitutions
- Command Substitution page 416

||, &&, and ; control operators
- Lists page 170
- ; and NEWLINE Separate Commands page 347
- && and || Boolean Control Operators page 349

## 2.2 USING THE COMMAND LINE TO GET HELP

### DESCRIPTION: RUNNING HELP COMMANDS AND NAVIGATION OF THE VARIOUS HELP SYSTEMS

**Key Knowledge Areas**

man

- man: Displays the System Manual page 135

info

- info: Displays Information About Utilities page 137

**Partial List of Used Files, Terms, and Utilities**

man

- man: Displays the System Manual page 135

info

- info: Displays Information About Utilities page 137

man pages

- man: Displays the System Manual page 135

**/usr/share/doc**

- **/usr/share/doc** page 141

locate

- locate: Searches for a File page 264

**Nice to Know**

apropos, whatis, whereis

- apropos: Searches for a Keyword page 137
- whatis page 137
- whereis page 263

## 2.3 USING DIRECTORIES AND LISTING FILES

### DESCRIPTION: NAVIGATION OF HOME AND SYSTEM DIRECTORIES AND LISTING FILES IN VARIOUS LOCATIONS

**Key Knowledge Areas**

Files, directories

- Ordinary Files and Directory Files page 185

Hidden files and directories

- Hidden Filenames page 188

Home

- Your Home Directory page 151

Absolute and relative paths

- Absolute Pathnames page 189
- Relative Pathnames page 190

**Partial List of Used Files, Terms, and Utilities**

Common options for ls

- Options page 229

Recursive listings

- Recursive page 230

cd

- cd: Changes to Another Working Directory page 193

. and ..

- The . and .. Directory Entries page 194

**home** and ~

- Your Home Directory page 151
- ~ (Tilde) in Pathnames page 190
- Tilde (~) page 365
- Tilde Expansion page 413

## 2.4 CREATING, MOVING, AND DELETING FILES

### DESCRIPTION: CREATE, MOVE, AND DELETE FILES AND DIRECTORIES UNDER THE HOME DIRECTORY

**Key Knowledge Areas**

Files and directories

- Ordinary Files and Directory Files page 185

Case sensitivity

- Case sensitivity page 187

Simple globbing and quoting

- Filename Generation/Pathname Expansion page 173
- Pathname Expansion page 418
- Special Characters page 150
- Quoting the $ page 394
- Quotation marks page 452



**Partial List of Used Files, Terms, and Utilities**

mv, cp, rm, touch

- mv, cp: Move or Copy Files page 195
- mv: Moves a Directory page 196
- mv: Renames or Moves a File page 245
- cp: Copies Files page 232
- rm: Removes a Link page 216
- rm: Removes a File (Deletes a Link) page 231
- touch: Changes File Modification and Access Times page 251

mkdir, rmdir

- mkdir: Creates a Directory page 192
- rmdir: Deletes a Directory page 194

# TOPIC 3: THE POWER OF THE COMMAND LINE

## 3.1 ARCHIVING FILES ON THE COMMAND LINE

### DESCRIPTION: ARCHIVING FILES IN THE USER HOME DIRECTORY

**Key Knowledge Areas**

Files, directories

- Ordinary Files and Directory Files page 185

Archives, compression

- Compressing and Archiving Files page 253

**Partial List of Used Files, Terms, and Utilities**

tar

- tar: Stores or Extracts Files to/from an Archive File page 257
- tar: Archives Files page 569

Common tar options

- Options page 257
- Modifiers page 259

gzip, bzip2

- xz, bzip2, and gzip: Compress and Decompress Files page 253

zip, unzip

- zip page 257
- unzip page 257

**Nice to Know**

Extracting individual files from archives

- Extract page 258

## 3.2 SEARCHING AND EXTRACTING DATA FROM FILES

### DESCRIPTION: SEARCH AND EXTRACT DATA FROM FILES IN THE HOME DIRECTORY

**Key Knowledge Areas**

Command line pipes

- Pipelines page 166

I/O redirection

- Redirection page 161

Partial POSIX Regular Expressions (., [ ], *, ?)

- Appendix A: Regular Expressions page 1161

**Partial List of Used Files, Terms, and Utilities**

find

- find: Finds Files Based on Criteria page 237

grep

- grep: Searches for a Pattern in Files page 240

less

- less Is more: Display a Text File One Screen at a Time page 228

head, tail

- head: Displays the Beginning of a File page 243

- tail: Displays the Last Part of a File page 249

sort
- sort: Sorts and/or Merges Files page 247

cut
- cut: Selects Characters or Fields from Input Lines page 233

wc
- wc: Displays the Number of Lines, Words, and Bytes in Files page 252

**Nice to Know**

Partial POSIX Basic Regular Expressions ([^ ], ^, $)
- Appendix A: Regular Expressions page 1161

Partial POSIX Extended Regular Expressions (+, ( ), |)
- Appendix A: Regular Expressions page 1161

xargs
- xargs: Converts Standard Input to Command Lines page 268

## 3.3 TURNING COMMANDS INTO A SCRIPT

### DESCRIPTION: TURNING REPETITIVE COMMANDS INTO SIMPLE SCRIPTS

**Key Knowledge Areas**

Basic text editing
- Tutorial: Using vim to Create and Edit a File page 270
- Tutorial: Using nano to Create and Edit a File page 277

Basic shell scripting
- Writing and Executing a Basic Shell Script page 134

**Partial List of Used Files, Terms, and Utilities**

**/bin/sh**
- sh Shell page 334

Variables
- Parameters and Variables page 358
- Variables page 1053

Arguments
- Arguments page 153

**for** loops
- **for...in** page 1017
- **for** page 1019

echo
- echo: Displays Arguments page 227
- **echo –e** page 1031

Exit status
- **$?**: Exit Status page 1051

**Nice to Know**

pico, nano, vi (only basics for creating scripts)
- Tutorial: Using vim to Create and Edit a File page 270

- Tutorial: Using nano to Create and Edit a File page 277
- pico, see Desktop Applications page 1173

bash
- Chapter 5: The Shell page 149
- Chapter 9: The Bourne Again Shell (bash) page 333
- Chapter 28: Programming the Bourne Again Shell (bash) page 1003

**if, while, case** statements
- **if...then** page 1005
- **if...then...else** page 1009
- **if...then...elif** page 1011
- **for...in** page 1017
- **for...in** page 1017

read and test, and [ commands
- test builtin page 1005
- [] is a synonym for test page 1008
- test builtin page 1022
- read: Accepts User Input page 1063

# TOPIC 4: THE LINUX OPERATING SYSTEM

## 4.1 CHOOSING AN OPERATING SYSTEM

### DESCRIPTION: KNOWLEDGE OF MAJOR OPERATING SYSTEMS AND LINUX DISTRIBUTIONS

**Key Knowledge Areas**

Windows, Mac, Linux differences
- Choosing an Operating System page 19

Distribution life cycle management

**Partial List of Used Files, Terms, and Utilities**

GUI versus command line, desktop configuration
- Choosing an Operating System page 19

Maintenance cycles, Beta and Stable
- beta release page 1235 (Glossary)
- stable release page 1274 (Glossary)

## 4.2 UNDERSTANDING COMPUTER HARDWARE

### DESCRIPTION: FAMILIARITY WITH THE COMPONENTS THAT GO INTO BUILDING DESKTOP AND SERVER COMPUTERS

**Key Knowledge Areas**

Hardware
- Requirements page 30

**Partial List of Used Files, Terms, and Utilities**

Hard drives and partitions, motherboards, processors, power supplies, optical drives, peripherals
- Processor Architecture page 32
- Setting Up the Hard Disk page 38

- Peripheral, see device page 1244 (Glossary)
- motherboard page 1260 (Glossary)
- optical drive page 1263 (Glossary)
- power supply page 1266 (Glossary)

Display types
- Interfaces: Installer and Installed System page 33
- Working from the Command Line page 125
- ASCII terminal page 1233 (Glossary)
- graphical display page 1249 (Glossary)

Drivers
- Device files page 494
- Block and Character Devices page 496
- device driver page 1244 (Glossary)

## 4.3 WHERE DATA IS STORED

### DESCRIPTION: WHERE VARIOUS TYPES OF INFORMATION ARE STORED ON A LINUX SYSTEM

**Key Knowledge Areas**

Kernel
- Linux kernel page 2
- Linux Is More Than a Kernel page 6
- Linux Has a Kernel Programming Interface page 11
- kernel page 1255 (Glossary)

Processes
- Process page 158
- Processes page 379
- ps page 456
- process page 1266 (Glossary)

syslog, klog, dmesg
- rsyslogd: Logs System Messages page 585
- **klogd**: deprecated; www.linuxjournal.com/article/4058
- dmesg: Displays Kernel Messages page 454

**/lib, /usr/lib, /etc, /var/log**
- **/etc** page 198
- **/lib** page 198
- **/lib64** page 198
- **/usr/lib** page 198
- **/usr/x86_64-linux-gnu** page 199
- **/var/log** page 199
- **/etc** page 483
- **/var/log** page 492
- Log Files and Mail for **root** page 590

**Partial List of Used Files, Terms, and Utilities**

Programs, libraries, packages and package databases, system configuration

- **/etc** page 198
- **/usr/bin** page 198
- **/lib** page 198
- **/lib64** page 198
- **/usr/lib** page 198
- **/usr/sbin** page 199
- **/usr/x86_64-linux-gnu** page 199
- **/etc** page 483
- library page 1256 (Glossary)
- Software package page 510
- PMS page 510
- Software package formats page 510
- Repositories page 511

Processes and process tables, memory addresses, system messaging, and logging

- Process page 158
- Processes page 379
- dmesg: Displays Kernel Messages page 454
- ps page 456
- **rsyslogd**: Logs System Messages page 585
- process page 1266 (Glossary)

ps, top, free

- free: Displays Memory Usage Information page 261
- Process Identification page 380
- ps page 456
- top: Lists Processes Using the Most Resources page 577



## 4.4 YOUR COMPUTER ON THE NETWORK

### DESCRIPTION: QUERYING VITAL NETWORKING SETTINGS AND DETERMINING THE BASIC REQUIREMENTS FOR A COMPUTER ON A LOCAL AREA NETWORK (LAN)

**Key Knowledge Areas**

Internet, network, routers

- Internet page 286
- Introduction to Networking page 286
- Internetworking Through Gateways and Routers page 293

Domain Name Service

- Chapter 25: DNS/BIND: Tracking Domain Names and Addresses page 891

Network configuration

- Chapter 16: Configuring and Monitoring a LAN page 661

**Partial List of Used Files, Terms, and Utilities**

route

- deprecated (route man page): see ip man page, **route** object instead

**resolv.conf**
- **/etc/resolv.conf** page 488

IPv4, IPv6
- IPv4 page 298
- IPv6 page 299

ifconfig
- deprecated: (ifconfig man page): see ip man page, **addr** and **link** objects instead

netstat
- netstat: see the netstat man page and wikipedia.org/wiki/netstat

ping
- ping: Tests a Network Connection page 311

**Nice to Know**

ssh
- ssh: Logs in or Executes Commands on a Remote System page 720

dig
- host and dig: Query Internet Nameservers page 313
- dig page 901
- dig page 902

# TOPIC 5: SECURITY AND FILE PERMISSIONS

## 5.1 BASIC SECURITY AND IDENTIFYING USER TYPES

### DESCRIPTION: VARIOUS TYPES OF USERS ON A LINUX SYSTEM

**Key Knowledge Areas**

Root and Standard Users
- Running Commands with **root** Privileges page 596
- The Special Powers of a User Working with **root** Privileges page 596
- Gaining **root** Privileges page 597
- Real UID Versus Effective UID page 599

System users
- **/etc/passwd** page 486

**Partial List of Used Files, Terms, and Utilities**

**/etc/passwd, /etc/group**
- **/etc/group** page 484
- **/etc/passwd** page 486

id, who, w
- w: Lists Users on the System page 262
- who: Lists Users on the System page 262
- who, whoami page 600
- id page 601

sudo
- Using sudo to Gain **root** Privileges page 602

**Nice to Know**

su

- Using su to Gain **root** Privileges page 600

## 5.2 CREATING USERS AND GROUPS

### DESCRIPTION: CREATING USERS AND GROUPS ON A LINUX SYSTEM

**Key Knowledge Areas**

User and group commands

- useradd: Adds a User Account page 566
- userdel: Removes a User Account page 566
- groupadd: Adds a Group page 567
- groupdel and groupmod: Remove and Modify a Group page 567
- usermod: Modifies a User Account page 567

User IDs

- /etc/passwd page 486
- Real UID Versus Effective UID page 599
- user ID page 1279 (Glossary)

**Partial List of Used Files, Terms, and Utilities**

**/etc/passwd, /etc/shadow, /etc/group**

- **/etc/group** page 484
- **/etc/passwd** page 486
- **/etc/shadow** page 489

id, last

- id page 601
- last: see the last man page

useradd, groupadd

- useradd: Adds a User Account page 566
- groupadd: Adds a Group page 567

passwd

- User Accounts: Changing Your Account Type and Password (GUI) page 118
- passwd: Changing Your Password (CLI) page 144

**Nice to Know**

usermod, userdel

- userdel: Removes a User Account page 566
- usermod: Modifies a User Account page 567

groupmod, groupdel

- groupdel and groupmod: Remove and Modify a Group page 567

## 5.3 MANAGING FILE PERMISSIONS AND OWNERSHIP

### DESCRIPTION: UNDERSTANDING AND MANIPULATING FILE PERMISSIONS AND OWNERSHIP SETTINGS

**Key Knowledge Areas**

File/directory permissions and owners

- Access Permissions page 199

**Partial List of Used Files, Terms, and Utilities**
**ls –l**
- **ls –l**: Displays Permissions page 199

chmod, chown
- chmod: Changes File Access Permissions page 201
- chown: Changes File Ownership page 203
- chmod: Makes a File Executable page 343

**Nice to Know**
chgrp
- chgrp: Changes File Group Association page 203

## 5.4 SPECIAL DIRECTORIES AND FILES
### DESCRIPTION: SPECIAL DIRECTORIES AND FILES ON A LINUX SYSTEM INCLUDING SPECIAL PERMISSIONS

**Key Knowledge Areas**
System files, libraries
- Important Standard Directories and Files page 197
- library page 1256 (Glossary)

Symbolic links
- Symbolic Links page 214
- Symbolic links page 494
- symbolic link page 1276 (Glossary)

**Partial List of Used Files, Terms, and Utilities**
**/etc, /var**
- **/etc** page 198
- **/etc** page 483
- **/var** page 43
- **/var** page 199

**/tmp, /var/tmp** and Sticky Bit
- **/tmp** page 198
- **/var** page 199
- Sticky bit page 204
- sticky bit page 1275 (Glossary)

**ls –d**
- Directory page 229

**ln –s**
- Size page 230

**Nice to Know**
Hard links
- Hard Links page 212

Setuid/Setgid
- Setuid and Setgid Permissions page 204
- Setuid file page 598
- Setuid files page 614
- setuid page 1271 (Glossary)
- setgid page 1271 (Glossary)

# CERTIFICATION EXAM 1 OBJECTIVES: LX0-101

## 101 SYSTEM ARCHITECTURE

### 101.1 DETERMINE AND CONFIGURE HARDWARE SETTINGS

Enable and disable integrated peripherals

Configure systems with or without external peripherals such as keyboards

Differentiate between the various types of mass storage devices

- **/dev** page 481

Set the correct hardware ID for different devices, especially the boot device

Know the differences between coldplug and hotplug devices

- Hotplug page 495

Determine hardware resources for devices

Tools and utilities to list various hardware information (e.g., lsusb, lspci, etc.)

- dmesg: Displays Kernel Messages page 454
- lspci: Lists PCI Information page 664
- lsblk: Lists Block Device Information page 665
- lshw: Lists Hardware Information page 665
- lsusb: Lists USB Devices page 666

Tools and utilities to manipulate USB devices

- Writing to a USB Flash Drive page 52

Conceptual understanding of sysfs, udev, hald, dbus

- udev page 494

**Partial List of Used Files, Terms, and Utilities**

**/sys**

- **/sys** page 198
- **/sys** page 492
- **/sys** page 495

**/proc**

- **/proc** page 198
- **/proc** page 490
- **proc** page 498

**/dev**

- Device file page 160
- **/dev** page 197
- **/dev** page 481
- Device files page 494

modprobe

- modprobe page 444

lsmod
- lsmod page 444

lspci
- lspci: Lists PCI Information page 664

lsusb
- lsusb: Lists USB Devices page 666

## 101.2 BOOT THE SYSTEM

Provide common commands to the boot loader and options to the kernel at boot time
- Modifying Boot Parameters (Options) page 75
- GRUB: The Linux Boot Loader page 444

Demonstrate knowledge of the boot sequence from BIOS to boot completion
- BIOS setup page 32
- CMOS page 32
- Booting the System page 438
- GRUB: The Linux Boot Loader page 444
- BIOS page 445
- BIOS page 1235 (Glossary)

Check boot events in the log file
- dmesg: Displays Kernel Messages page 454

**Partial List of Used Files, Terms, and Utilities**

**/var/log/messages**
- **/var/log/messages** page 492
- **/var/log/syslog** page 587
- Log Files and Mail for **root** page 590
- Truncating log files page 591

dmesg
- dmesg: Displays Kernel Messages page 454

BIOS
- BIOS setup page 32
- BIOS page 445
- BIOS page 1235 (Glossary)

boot loader
- GRUB: The Linux Boot Loader page 444

kernel
- Linux kernel page 2
- Linux Is More Than a Kernel page 6
- Linux Has a Kernel Programming Interface page 11
- kernel page 1255 (Glossary)

**init**
- **init** daemon page 380
- The Upstart Event-Based **init** Daemon page 427
- SysVinit (**rc**) Scripts: Start and Stop System Services page 435
- Upstart **init** daemon page 439

## 101.3 CHANGE RUNLEVELS AND SHUTDOWN OR REBOOT SYSTEM

Set the default runlevel
- **rc-sysinit** task and **inittab** page 435
- **/etc/inittab** page 486

Change between runlevels including single-user mode
- telinit page 438
- Going to Graphical Multiuser Mode page 439
- Booting the System to Recovery (Single-User) Mode page 451

Shutdown and reboot from the command line
- Bringing the System Down page 441

Alert users before switching runlevels or other major system events

Properly terminate processes
- kill: Aborting a Background Job page 172
- kill: Sends a Signal to a Process page 455
- killall: Kills a Command page 457
- pkill: Kills a Command page 458
- kill: Aborts a Process page 1072

**Partial List of Used Files, Terms, and Utilities**

**/etc/inittab**
- **rc-sysinit** task and **inittab** page 435
- **/etc/inittab** page 486

shutdown
- Bringing the System Down page 441

**init**
- **init** daemon page 380
- The Upstart Event-Based **init** Daemon page 427
- SysVinit (**rc**) Scripts: Start and Stop System Services page 435
- Upstart **init** daemon page 439

**/etc/init.d**
- SysVinit (**rc**) Scripts: Start and Stop System Services page 435

telinit
- telinit page 438



# 102 LINUX INSTALLATION AND PACKAGE MANAGEMENT

## 102.1 DESIGN HARD DISK LAYOUT

Allocate filesystems and swap space to separate partitions or disks
- Setting Up the Hard Disk page 38

Tailor the design to the intended use of the system
- Planning the Installation page 30

Ensure the **/boot** partition conforms to the hardware architecture requirements for booting
- Where to put the **/boot** partition page 43
- LBA mode and the **/boot** partition page 445

**Partial List of Used Files, Terms, and Utilities**

**/** (root) filesystem
- **/** (root) page 42
- **/** (root) page 189
- **/** page 197
- root filesystem page 1270 (Glossary)

**/var** filesystem
- **/var** page 43
- **/var** page 199

**/home** filesystem
- **/home** page 44
- **/home** page 198

swap space
- (swap) page 42
- swap page 491
- swap space page 1276 (Glossary)

mount points
- Mount Points page 40
- Mount point page 499

partitions
- Partitions page 38
- Partition table page 38
- Primary, Extended, and Logical Partitions page 39
- Guided Partitioning page 41
- Guided partitioning page 65
- The ubiquity Advanced Partitioning Screen page 67
- Manual Partitioning Using the Textual Partition Editor page 82
- gnome-disks: The GNOME Disk Utility page 88
- Example minimum partition sizes page 44
- partition page 1264 (Glossary)

## 102.2 INSTALL A BOOT MANAGER

Providing alternative boot locations and backup boot options

Install and configure a boot loader such as GRUB
- GRUB: The Linux Boot Loader page 444

Interact with the boot loader
- Booting the System to Recovery (Single-User) Mode page 451

**Partial List of Used Files, Terms, and Utilities**

**/boot/grub/menu.lst**
- Configuring GRUB page 445

grub-install
- grub-install: Installs the MBR and GRUB Files page 450

MBR
- Reinstalling the MBR page 450
- MBR page 445
- grub-install: Installs the MBR and GRUB Files page 450

superblock
- superblock page 1275 (Glossary)

**/etc/lilo.conf**

lilo deprecated

## 102.3 MANAGE SHARED LIBRARIES

Identify shared libraries
- ldd page 618

Identify the typical locations of system libraries

Load shared libraries

**Partial List of Used Files, Terms, and Utilities**

ldd
- ldd & **libwrap** page 616
- ldd page 618

ldconfig

**/etc/ld.so.conf**

**LD_LIBRARY_PATH**

## 102.4 USE DEBIAN PACKAGE MANAGEMENT

Install, upgrade, and uninstall Debian binary packages
- JumpStart: Installing and Removing Software Packages Using apt-get page 512

Find packages containing specific files or libraries which may or may not be installed
- Finding the Package That Holds an Application or File You Need page 514

Obtain package information like version, content, dependencies, package integrity, and installation status (whether or not the package is installed)
- apt-cache: Displays Package Information page 522
- **dpkg —list**: Displays Information About a Package page 526

**Partial List of Used Files, Terms, and Utilities**

**/etc/apt/sources.list**
- **sources.list**: Specifies Repositories for APT to Search page 516

dpkg
- dpkg: The Debian Package Management System page 524

dpkg-reconfigure
- dpkg-reconfigure: Reconfigures **postfix** page 796

apt-get
- JumpStart: Installing and Removing Software Packages Using apt-get page 512
- apt-get: Works with Packages and the Local Package Index page 519
- apt-get source: Downloads Source Files page 523

apt-cache
- apt-cache: Displays Package Information page 522

aptitude

## 102.5 USE RPM AND YUM PACKAGE MANAGEMENT

*See Chapter 12: Finding, Downloading, and Installing Software* page 509

Install, re-install, upgrade, and remove packages using RPM and YUM
- Installing and Removing Software Packages Using yum page 1178
- Working with yum page 1179

Obtain information on RPM packages such as version, status, dependencies, integrity, and signatures
- Working with yum page 1179

Determine what files a package provides, as well as find which package a specific file comes from
- Finding the Package That Holds a File You Need page 1179

**Partial List of Used Files, Terms, and Utilities**

rpm
- RPM page 1178

rpm2cpio

**/etc/yum.conf**
- **yum.conf**: Configures yum page 1182

**/etc/yum.repos.d/**
- yum Repositories page 1182

yum
- Installing and Removing Software Packages Using yum page 1178
- Working with yum page 1179
- Finding the Package That Holds a File You Need page 1179
- Updating Packages page 1180
- yum Commands page 1181
- **yum.conf**: Configures yum page 1182
- yum Repositories page 1182

yumdownloader



# 103 GNU AND UNIX COMMANDS

## 103.1 WORK ON THE COMMAND LINE

*See Chapter 5: The Shell page 149*

*See Chapter 9: The Bourne Again Shell (bash) page 333*

*See Chapter 7: The Linux Utilities page 223*

*See Chapter 28: Programming the Bourne Again Shell (bash) page 1003*

Use single shell commands and one line command sequences to perform basic tasks on the command line
- Chapter 5: The Shell page 149
- Chapter 7: The Linux Utilities page 223
- Chapter 9: The Bourne Again Shell (bash) page 333

Use and modify the shell environment including defining, referencing, and exporting environment variables

- Parameters and Variables page 358
- Variables page 1053

Use and edit command history

- History page 382

Invoke commands inside and outside the defined path

- Absolute versus relative pathnames page 157
- **PATH**: Where the Shell Looks for Programs page 365

**Partial List of Used Files, Terms, and Utilities**

**.** (dot)

- **.** (Dot) or source: Runs a Startup File in the Current Shell page 338
- exec versus **.** (dot) page 1067

bash

- Chapter 5: The Shell page 149
- Chapter 9: The Bourne Again Shell (bash) page 333
- Chapter 28: Programming the Bourne Again Shell (bash) page 1003

echo

- echo: Displays Arguments page 227
- **echo –e** page 1031

env

- env: Runs a Program in a Modified Environment page 1057

exec

- Opening a File Descriptor page 1039
- Duplicating a File Descriptor page 1039
- exec: Executes a Command or Redirects File Descriptors page 1067

export

- declare: Lists and Assigns Attributes to Variables page 363
- readonly and export page 363
- export: Puts Variables in the Environment page 1054

pwd

- pwd page 151

set

- **set ±o**: Turns Shell Features On and Off page 406
- set: Initializes Positional Parameters page 1046

unset

- unset: Removes a Variable page 362

man

- man: Displays the System Manual page 135

uname

- uname: Displays System Information page 460

history
- History page 382

## 103.2 PROCESS TEXT STREAMS USING FILTERS

Send text files and output streams through text utility filters to modify the output using standard UNIX commands found in the GNU **textutils** package
- Redirection page 161
- Pipelines page 166
- Filters page 169

**Partial List of Used Files, Terms, and Utilities**

cat
- cat: Joins and Displays Files page 224
- cat page 160
- Redirection page 161

cut
- cut: Selects Characters or Fields from Input Lines page 233

expand

fmt

head
- head: Displays the Beginning of a File page 243

od

join

nl

paste

pr

sed

sort
- sort: Sorts and/or Merges Files page 247

split

tail
- tail: Displays the Last Part of a File page 249

tr
- tr page 167
- tr page 268

unexpand

uniq

wc
- wc: Displays the Number of Lines, Words, and Bytes in Files page 252

## 103.3 PERFORM BASIC FILE MANAGEMENT

Copy, move, and remove files and directories individually
- rmdir: Deletes a Directory page 194

- mv, cp: Move or Copy Files page 195
- mv: Moves a Directory page 196
- rm: Removes a Link page 216
- rm: Removes a File (Deletes a Link) page 231
- cp: Copies Files page 232
- mv: Renames or Moves a File page 245

Copy multiple files and directories recursively
- cp: Copies Files page 232

Remove files and directories recursively
- rm: Removes a File (Deletes a Link) page 231

Use simple and advanced wildcard specifications in commands
- Filename Generation/Pathname Expansion page 173
- Pathname Expansion page 418

Using find to locate and act on files based on type, size, or time
- find: Finds Files Based on Criteria page 237

Usage of tar, cpio, and dd
- tar: Stores or Extracts Files to/from an Archive File page 257
- tar: Archives Files page 569
- cpio: Archives Files page 571

**Partial List of Used Files, Terms, and Utilities**

cp
- mv, cp: Move or Copy Files page 195
- cp: Copies Files page 232

find
- find: Finds Files Based on Criteria page 237

mkdir
- mkdir: Creates a Directory page 192

mv
- mv, cp: Move or Copy Files page 195
- mv: Moves a Directory page 196
- mv: Renames or Moves a File page 245

ls
- **ls –l**: Displays Permissions page 199
- ls: Displays Information About Files page 229

rm
- rm: Removes a Link page 216
- rm: Removes a File (Deletes a Link) page 231

rmdir
- rmdir: Deletes a Directory page 194

touch
- touch: Changes File Modification and Access Times page 251

tar
- tar: Stores or Extracts Files to/from an Archive File page 257
- tar: Archives Files page 569

cpio
- cpio: Archives Files page 571

dd

file
- file: Displays the Classification of a File page 237

gzip
- xz, bzip2, and gzip: Compress and Decompress Files page 253

gunzip
- unxz bunzip2 gunzip page 256

bzip2
- xz, bzip2, and gzip: Compress and Decompress Files page 253

file globbing
- Filename Generation/Pathname Expansion page 173
- Pathname Expansion page 418

## 103.4 USE STREAMS, PIPES, AND REDIRECTS

Redirecting standard input, standard output, and standard error
- Redirecting Standard Output page 162
- Redirecting Standard Input page 163
- Redirecting Standard Error page 339
- redirection page 1268 (Glossary)
- standard error page 1274 (Glossary)
- standard input page 1274 (Glossary)
- standard output page 1274 (Glossary)

Pipe the output of one command to the input of another command
- Pipelines page 166
- Filters page 169
- filter page 1248 (Glossary)
- pipeline page 1265 (Glossary)

Use the output of one command as arguments to another command
- xargs: Converts Standard Input to Command Lines page 268

Send output to both stdout and a file
- tee page 170

**Partial List of Used Files, Terms, and Utilities**

tee
- tee page 170

xargs
- xargs: Converts Standard Input to Command Lines page 268

## 103.5 CREATE, MONITOR, AND KILL PROCESSES

Run jobs in the foreground and background

- Running a Command in the Background page 171
- Moving a Job from the Foreground to the Background page 172
- Background process page 381
- background process page 1235 (Glossary)
- foreground process page 1248 (Glossary)

Signal a program to continue running after logout

Monitor active processes

- Process Identification page 380
- ps page 380
- ps page 456

Select and sort processes for display

- Process Identification page 380
- ps page 456
- top: Lists Processes Using the Most Resources page 577

Send signals to processes

- Aborting Execution page 130
- kill: Aborting a Background Job page 172
- kill: Sends a Signal to a Process page 455
- killall: Kills a Command page 457
- pkill: Kills a Command page 458
- Signals page 1069

**Partial List of Used Files, Terms, and Utilities**

&

- Running a Command in the Background page 171
- Background process page 381
- background process page 1235 (Glossary)
- foreground process page 1248 (Glossary)

bg

- Moving a Job from the Foreground to the Background page 172
- bg: Sends a Job to the Background page 354
- background process page 1235 (Glossary)

fg

- Foreground page 171
- Moving a Job from the Foreground to the Background page 172
- fg: Brings a Job to the Foreground page 353
- foreground process page 1248 (Glossary)

jobs

- Determining the number of a job using jobs page 172
- jobs: Lists Jobs page 352

kill

- kill: Aborting a Background Job page 172
- kill: Sends a Signal to a Process page 455

nohup

ps

- Process Identification page 380
- ps page 456

top

- top: Lists Processes Using the Most Resources page 577

free

- free: Displays Memory Usage Information page 261

uptime

- uptime: Displays System Load and Duration Information page 261

killall

- killall: Kills a Command page 457

## 103.6 MODIFY PROCESS EXECUTION PRIORITIES

Know the default priority of a job that is created

- Process Identification page 380
- ps page 456
- top: Lists Processes Using the Most Resources page 577

Run a program with higher or lower priority than the default

Change the priority of a running process

**Partial List of Used Files, Terms, and Utilities**

nice

ps

- Process Identification page 380
- ps page 456

renice

top

- top: Lists Processes Using the Most Resources page 577

## 103.7 SEARCH TEXT FILES USING REGULAR EXPRESSIONS

*See Appendix A: Regular Expressions page 1161*

Create simple regular expressions containing several notational elements

- Appendix A: Regular Expressions page 1161
- Regular Expressions page 1123 (Python)

Use regular expression tools to perform searches through a filesystem or file content

- *See preceding entry.*

**Partial List of Used Files, Terms, and Utilities**

grep

- grep: Searches for a Pattern in Files page 240

egrep

- Extended regular expression page 241

fgrep

sed

- **regex**(7)

## 103.8 PERFORM BASIC FILE EDITING OPERATIONS USING vi

*A Practical Guide to Ubuntu Linux, Fourth Edition covers the* vim *editor. All commands discussed here are compatible between* vi *and* vim.

Tutorial: Using vim to Create and Edit a File page 270

Navigate a document using vi

- Moving the Cursor page 274

Use basic vi modes

- Command and Input Modes page 272

Insert, edit, delete, copy, and find text

- Entering Text page 273
- Deleting Text page 275
- Correcting Text page 276

**Partial List of Used Files, Terms, and Utilities**

vi

- Tutorial: Using vim to Create and Edit a File page 270

**/, ?**

**h, j, k, l**

- Moving the Cursor page 274

**i, o, a**

- Entering Text page 273
- Entering Additional Text page 276

**c, d, p, y, dd, yy**

- Deleting Text page 275

**ZZ, :w!, :q!, :e!**

- Ending the Editing Session page 276



# 104 DEVICES, LINUX FILESYSTEMS, FILESYSTEM HIERARCHY STANDARD

## 104.1 CREATE PARTITIONS AND FILESYSTEMS

Use various mkfs commands to set up partitions and create various filesystems such as:

**ext2**

- **ext2** page 497
- **ext2** to **ext3** page 505

**ext3**

- **ext3** page 497
- **ext3** to **ext2** page 505

**xfs**

- The XFS Filesystem page 506

**reiserfs v3**

- **reiserfs** page 498

**vfat**

- **vfat** page 498

**Partial List of Used Files, Terms, and Utilities**

fdisk

- fdisk: see the fdisk man page
- See also parted: Reports on and Partitions a Hard Disk page 579

mkfs

- mkfs: Creates a Filesystem page 457

mkswap

- **swap** page 491

## 104.2 MAINTAIN THE INTEGRITY OF FILESYSTEMS

Verify the integrity of filesystems

- fsck: Checks Filesystem Integrity page 503

Monitor free space and inodes

- df: shows where directory hierarchies are mounted page 846

Repair simple filesystem problems

- fsck: Checks Filesystem Integrity page 503

**Partial List of Used Files, Terms, and Utilities**

du

- du: Displays Disk Usage Information page 501

df

- df: shows where directory hierarchies are mounted page 846

fsck

- fsck: Checks Filesystem Integrity page 503

e2fsck

mke2fs

debugfs

dumpe2fs

tune2fs

- tune2fs: Changes Filesystem Parameters page 504

xfs tools (such as xfs_metadump and xfs_info)

## 104.3 CONTROL MOUNTING AND UNMOUNTING OF FILESYSTEMS

Manually mount and unmount filesystems

- mount: Mounts a Filesystem page 499
- umount: Unmounts a Filesystem page 501
- mount: Mounts a Directory Hierarchy page 849
- Mounting Shares page 876

Configure filesystem mounting on bootup

- **fstab**: Keeps Track of Filesystems page 502
- **fstab** file page 849
- **/etc/fstab**: Mounts Directory Hierarchies Automatically page 853

Configure user mountable removable filesystems
- Mount Options page 500

**Partial List of Used Files, Terms, and Utilities**
**/etc/fstab**
- **fstab**: Keeps Track of Filesystems page 502
- **fstab** file page 849
- **/etc/fstab**: Mounts Directory Hierarchies Automatically page 853

**/media**

mount
- mount: Mounts a Filesystem page 499
- mount: Mounts a Directory Hierarchy page 849
- Mounting Shares page 876

umount
- umount: Unmounts a Filesystem page 501

## 104.4 MANAGE DISK QUOTAS

Set up a disk quota for a filesystem
- Disk Quota System page 592

Edit, check and generate user quota reports
- quota and repquota page 593

**Partial List of Used Files, Terms, and Utilities**
quota
- quota and repquota page 593

edquota
- edquota and quotaon page 593

repquota
- quota and repquota page 593

quotaon
- edquota and quotaon page 593



## 104.5 MANAGE FILE PERMISSIONS AND OWNERSHIP

Manage access permissions on regular and special files as well as directories
- chmod: Changes File Access Permissions page 201
- chmod: Makes a File Executable page 343

Use access modes such as suid, sgid, and the sticky bit to maintain security
- Setuid and Setgid Permissions page 204
- Setuid file page 598
- Setuid files page 614
- setgid page 1271 (Glossary)
- setuid page 1271 (Glossary)

Know how to change the file creation mask
- umask: Specifies the File Permission Mask page 459

Use the group field to grant file access to group members
- **ls –l**: Displays Permissions page 199
- chmod: Changes File Access Permissions page 201
- **/etc/group** page 484

**Partial List of Used Files, Terms, and Utilities**

chmod
- chmod: Changes File Access Permissions page 201
- chmod: Makes a File Executable page 343

umask
- umask: Specifies the File Permission Mask page 459

chown
- chown: Changes File Ownership page 203

chgrp
- chgrp: Changes File Group Association page 203

## 104.6 CREATE AND CHANGE HARD AND SYMBOLIC LINKS

Create links
- ln: Creates a Hard Link page 212
- ln: Creates Symbolic Links page 215

Identify hard and/or softlinks
- ls and link counts page 214
- ls and inodes page 214
- hard link page 1250 (Glossary)
- link page 1256 (Glossary)
- symbolic link page 1276 (Glossary)

Copying versus linking files
- cp Versus ln page 213

Use links to support system administration tasks
- ln: Creates a Hard Link page 212
- ln: Creates Symbolic Links page 215

**Partial List of Used Files, Terms, and Utilities**

ln
- ln: Creates a Hard Link page 212
- ln: Creates Symbolic Links page 215

## 104.7 FIND SYSTEM FILES AND PLACE FILES IN THE CORRECT LOCATION

Understand the correct locations of files under the FHS
- Important Standard Directories and Files page 197
- Important Files and Directories page 480

Find files and commands on a Linux system
- whereis page 263
- locate: Searches for a File page 264

Know the location and purpose of important files and directories as defined in the FHS
- Important Standard Directories and Files page 197
- Important Files and Directories page 480

**Partial List of Used Files, Terms, and Utilities**

find
- find: Finds Files Based on Criteria page 237

locate
- locate: Searches for a File page 264

updatedb
- updatedb page 264

whereis
- whereis page 263

which
- which page 263

type
- type: Displays Information About a Command page 1063

**/etc/updatedb.conf**

# CERTIFICATION EXAM 2 OBJECTIVES: LX0-102

## 105 SHELLS, SCRIPTING, AND DATA MANAGEMENT

### 105.1 CUSTOMIZE AND USE THE SHELL ENVIRONMENT

*See Chapter 9: The Bourne Again Shell (bash)* page 333

Set environment variables (e.g., **PATH**) at login or when spawning a new shell
- Startup Files page 335
- Set **PATH** in **.bash_profile** page 337
- Keyword variables page 359
- Keyword Variables page 364

Write bash functions for frequently used sequences of commands
- Functions page 402
- Variables in Functions page 1061

Maintain skeleton directories for new user accounts
- useradd: Adds a User Account page 566

Set command search path with the proper directory
- **PATH**: Where the Shell Looks for Programs page 365

**Partial List of Used Files, Terms, and Utilities**

**/etc/profile**
- **/etc/profile** page 336
- **/etc/profile** and **/etc/profile.d** page 487

env
- env: Runs a Program in a Modified Environment page 1057

export
- declare: Lists and Assigns Attributes to Variables page 363
- export: Puts Variables in the Environment page 1054

set
- **set ±o**: Turns Shell Features On and Off page 406
- set: Initializes Positional Parameters page 1046

unset
- unset: Removes a Variable page 362

**~/.bash_profile**
- **.bash_profile**, **.bash_login**, and **.profile** page 336
- **~/.bash_profile** page 480

**~/.bash_login**
- **.bash_profile**, **.bash_login**, and **.profile** page 336

**~/.profile**
- **.bash_profile**, **.bash_login**, and **.profile** page 336

**~/.bashrc**
- **.bashrc** page 337
- **~/.bashrc** page 480

**~/.bash_logout**
- **.bash_logout** page 336

Functions
- Functions page 402
- Variables in Functions page 1061

Aliases
- Aliases page 398
- Alias Substitution page 410

lists
- Lists page 170

## 105.2 CUSTOMIZE OR WRITE SIMPLE SCRIPTS

*See Chapter 9: The Bourne Again Shell (bash)* page 333

*See Chapter 28: Programming the Bourne Again Shell (bash)* page 1003

Use standard sh syntax (loops, tests)
- Control Structures page 1004

Use command substitution
- Command Substitution page 416

Test return values for success or failure or other information provided by a command
- test builtin page 1005
- [] is a synonym for test page 1008
- test builtin page 1022

Perform conditional mailing to the superuser

Correctly select the script interpreter through the shebang (#!) line

- **#!** Specifies a Shell page 344

Manage the location, ownership, execution, and suid-rights of scripts

- Listing setuid files page 614

**Partial List of Used Files, Terms, and Utilities**

**for**

- **for...in** page 1017
- **for** page 1019

**while**

- **while** page 1021

test

- test builtin page 1005
- [] is a synonym for test page 1008
- test builtin page 1022

**if**

- **if...then** page 1005
- **if...then...else** page 1009
- **if...then...elif** page 1011

read

- read: Accepts User Input page 1063

seq

- seq page 413

## 105.3 SQL DATA MANAGEMENT

*See Chapter 30: The MariaDB SQL Database Management System page 1135*

Use of basic SQL commands

- Notes page 1136

Perform basic data manipulation

- Examples page 1145

**Partial List of Used Files, Terms, and Utilities**

insert

- INSERT INTO page 1147

update

- UPDATE page 1150

select

- Retrieving Data page 1148
- Joins page 1152

delete

- DELETE FROM page 1150

from

- DELETE FROM page 1150

where
- ▸ WHERE page 1149

group by

order by
- ▸ ORDER BY page 1148

join
- ▸ Joins page 1152

# 106 USER INTERFACES AND DESKTOPS

## 106.1 INSTALL AND CONFIGURE X11

*See X Window System page 471*

Verify that the video card and monitor are supported by an X server
- • Displays page 116

Awareness of the X font server

Basic understanding and knowledge of the X Window configuration file

**Partial List of Used Files, Terms, and Utilities**
**/etc/X11/xorg.conf**

xhost
- ▸ xhost Grants Access to a Display page 473

**DISPLAY**
- ▸ The **DISPLAY** Variable page 474

xwininfo

xdpyinfo

X
- ▸ X Window System page 471

## 106.2 SET UP A DISPLAY MANAGER

Turn the display manager on or off

Change the display manager greeting

Change default color depth for the display manager

Configure display managers for use by X-stations

**Partial List of Used Files, Terms, and Utilities**
**/etc/inittab**
- ▸ **/etc/inittab** page 486

**xdm** configuration files

**kdm** configuration files

**gdm** configuration files
- ▸ Graphical login page 440
- ▸ The **Xorg –nolisten tcp** Option page 472

## 106.3 ACCESSIBILITY

Keyboard Accessibility Settings (AccessX?)
Visual Settings and Themes
Assistive Technology (ATs)

**Partial List of Used Files, Terms, and Utilities**
Sticky/Repeat Keys
Slow/Bounce/Toggle Keys
Mouse Keys
High Contrast/Large Print Desktop Themes
Screen Reader
Braille Display
Screen Magnifier
On-Screen Keyboard
Gestures (used at login, for example gdm)
Orca
GOK
emacspeak

# 107 ADMINISTRATIVE TASKS

## 107.1 MANAGE USER AND GROUP ACCOUNTS AND RELATED SYSTEM FILES

Add, modify, and remove users and groups
- unity-control-center: Manages User Accounts page 564
- Managing User Accounts from the Command Line page 566

Manage user/group info in password/group databases
- Modifying a User page 566
- usermod: Modifies a User Account page 567
- groupadd: Adds a Group page 567
- groupdel and groupmod: Remove and Modify a Group page 567
- chage page 567

Create and manage special purpose and limited accounts

**Partial List of Used Files, Terms, and Utilities**
**/etc/passwd**
- **/etc/passwd** page 486

**/etc/shadow**
- **/etc/shadow** page 489

**/etc/group**
- **/etc/group** page 484

**/etc/skel**
- **/etc/skel** page 566

chage
- chage page 567

groupadd
- groupadd: Adds a Group page 567

groupdel
- groupdel and groupmod: Remove and Modify a Group page 567

groupmod
- groupdel and groupmod: Remove and Modify a Group page 567

passwd
- User Accounts: Changing Your Account Type and Password (GUI) page 118
- passwd: Changing Your Password (CLI) page 144

useradd
- useradd: Adds a User Account page 566

userdel
- userdel: Removes a User Account page 566

usermod
- usermod: Modifies a User Account page 567

## 107.2 AUTOMATE SYSTEM ADMINISTRATION TASKS BY SCHEDULING JOBS

Manage **cron** and at jobs
- cron and anacron: Schedule Routine Tasks page 573
- at: Runs Occasional Tasks page 576

Configure user access to **cron** and at services
- **/etc/at.allow**, **/etc/at.deny**, **/etc/cron.allow**, and **/etc/cron.deny** page 484

**Partial List of Used Files, Terms, and Utilities**

**/etc/cron.{d,daily,hourly,monthly,weekly}**
- Crontab Files page 573

**/etc/at.deny**
- **/etc/at.allow**, **/etc/at.deny**, **/etc/cron.allow**, and **/etc/cron.deny** page 484

**/etc/at.allow**
- **/etc/at.allow**, **/etc/at.deny**, **/etc/cron.allow**, and **/etc/cron.deny** page 484

**/etc/crontab**
- **/etc/crontab** page 574

**/etc/cron.allow**
- **/etc/at.allow**, **/etc/at.deny**, **/etc/cron.allow**, and **/etc/cron.deny** page 484

**/etc/cron.deny**
- **/etc/at.allow**, **/etc/at.deny**, **/etc/cron.allow**, and **/etc/cron.deny** page 484

**/var/spool/cron/***
- Crontab Files page 573

crontab
- User crontab files page 574
- cron and anacron: Schedule Routine Tasks page 573

at
- at: Runs Occasional Tasks page 576

atq

atrm

## 107.3 LOCALIZATION AND INTERNATIONALIZATION

Locale settings
- Locale page 374
- locale page 1257 (Glossary)

Time zone settings
- tzconfig page 378
- tzselect page 378
- **/etc/timezone** page 378

**Partial List of Used Files, Terms, and Utilities**

**/etc/timezone**
- **/etc/timezone** page 378

**/etc/localtime**
- **/etc/localtime** page 379

**/usr/share/zoneinfo**
- **/usr/share/zoneinfo** page 378

Environment variables:
- **LC_**: Locale Variables page 374
- Environment Variables page 1054

**/usr/bin/locale**
- locale: Displays Locale Information page 375

tzselect
- tzselect page 378

tzconfig
- tzconfig page 378

date
- date: Displays the System Time and Date page 226

iconv

UTF-8
- **LC_**: Locale Variables page 374
- UTF-8 page 1280 (Glossary)

ISO-8859
- LC_: Locale Variables page 374

ASCII
- ASCII page 1233 (Glossary)

Unicode
- Unicode page 1279 (Glossary)

# 108 ESSENTIAL SYSTEM SERVICES

## 108.1 MAINTAIN SYSTEM TIME

Set the system date and time
- timedatectl: Reports on and Sets the System Clock page 579

Set the hardware clock to the correct time in UTC

Configure the correct time zone
- Time page 377

Basic NTP configuration

Knowledge of using the pool.ntp.org service

**Partial List of Used Files, Terms, and Utilities**

**/usr/share/zoneinfo**
- **/usr/share/zoneinfo** page 378

**/etc/timezone**
- **/etc/timezone** page 378

**/etc/localtime**
- **/etc/localtime** page 379

**/etc/ntp.conf**

date
- date: Displays the System Time and Date page 226
- timedatectl: Reports on and Sets the System Clock page 579

hwclock

**ntpd**

ntpdate

**pool.ntp.org**

## 108.2 SYSTEM LOGGING

Syslog configuration files
- **rsyslog.conf** page 585

**syslog**
- **rsyslogd**: Logs System Messages page 585

standard facilities, priorities, and actions
- Selectors page 585
- Facilities page 585
- Priorities page 586
- Actions page 586

**Partial List of Used Files, Terms, and Utilities**
**syslog.conf**
- **rsyslog.conf** page 585

**syslogd**
- **rsyslogd**: Logs System Messages page 585

**klogd**

logger

## 108.3 MAIL TRANSFER AGENT (MTA) BASICS

*See Chapter 21: **postfix**: Setting Up Mail Servers, Clients, and More* page 779

Create e-mail aliases
- **/etc/aliases** page 794

Configure e-mail forwarding
- **~/.forward** page 796

Knowledge of commonly available MTA programs (**postfix, sendmail,** Qmail, **exim**) (no configuration)
- Alternatives to **postfix** page 783

**Partial List of Used Files, Terms, and Utilities**
**~/.forward**
- **~/.forward** page 796

**sendmail** emulation layer commands

newaliases
- newaliases page 795

mail

mailq
- mailq page 786

**postfix**
- Introduction to **postfix** page 781
- Setting Up a **postfix** Mail Server page 784
- JumpStart: Configuring **postfix** to Use Gmail as a Smarthost page 787
- Configuring **postfix** page 789

**sendmail**
- The **postfix** to **sendmail** Compatibility Interface page 782
- **sendmail** page 783
- Outbound Email page 785

**exim**
- **exim4** page 783

**qmail**
- Qmail page 783

## 108.4 MANAGE PRINTERS AND PRINTING

*See Chapter 13: Printing with CUPS* page 539

Basic CUPS configuration (for local and remote printers)
- The System Configures a Local Printer Automatically page 542
- JumpStart I: Configuring a Printer Using system-config-printer page 542
- JumpStart II: Setting Up a Local or Remote Printer page 544
- Working with the CUPS Web Interface page 548
- Configuring Printers page 549

Manage user print queues
- Managing Print Queues page 555

Troubleshoot general printing problems

Add and remove jobs from configured printer queues
- BSD and System V command-line print utilities page 557

**Partial List of Used Files, Terms, and Utilities**

CUPS configuration files, tools, and utilities
- JumpStart I: Configuring a Printer Using system-config-printer page 542
- Working with the CUPS Web Interface page 548
- Sharing CUPS Printers page 555

**/etc/cups**
- Example lpadmin Commands page 553

**lpd** legacy interface (lpr, lprm, lpq)
- Traditional UNIX Printing page 557



# 109 NETWORKING FUNDAMENTALS

*See Chapter 8: Networking and the Internet* page 285

## 109.1 FUNDAMENTALS OF INTERNET PROTOCOLS

*See Network Protocols* page 296

Demonstrate an understanding of network masks
- Subnet mask page 304
- network mask page 1262 (Glossary)

Knowledge of the differences between private and public "dotted quad" IP addresses
- Private address space page 667
- private address space page 1266 (Glossary)

Setting a default route

Knowledge about common TCP and UDP ports (20, 21, 22, 23, 25, 53, 80, 110, 119, 139, 143, 161, 443, 465, 993, 995)
- Each chapter covering a server discusses which ports that server uses.

- Ports page 318
- port page 1265 (Glossary)

Knowledge about the differences and major features of UDP, TCP, and ICMP

- UDP page 296
- UDP: User Datagram Protocol page 298
- TCP page 296
- TCP: Transmission Control Protocol page 297
- ping: Tests a Network Connection page 311
- UDP page 1279 (Glossary)
- TCP page 1276 (Glossary)
- ICMP page 1252 (Glossary)

Knowledge of the major differences between IPv4 and IPv6

- IPv4 page 298
- IPv6 page 299

**Partial List of Used Files, Terms, and Utilities**

**/etc/services**

- Network Services page 319
- **/etc/services** page 489

ftp

- Chapter 20: FTP: Transferring Files Across a Network page 753

telnet

- telnet: Logs In on a Remote System page 309

host

- host and dig: Query Internet Nameservers page 313

ping

- ping: Tests a Network Connection page 311

dig

- host and dig: Query Internet Nameservers page 313
- dig page 901
- dig page 902

traceroute

- traceroute: Traces a Route over the Internet page 312

tracepath

## 109.2 BASIC NETWORK CONFIGURATION

Manually and automatically configure network interfaces

- Configuring the Systems page 666
- NetworkManager: Configures Network Connections page 667

Basic TCP/IP host configuration

**Partial List of Used Files, Terms, and Utilities**

**/etc/hostname**

- **/etc/hostname** page 485

**/etc/hosts**

- Hostnames page 306
- **/etc/hosts** page 485

**/etc/resolv.conf**

- **/etc/resolv.conf** page 488

**/etc/nsswitch.conf**

- nsswitch.conf: Which Service to Look at First page 468

ifconfig

ifup

ifdown

route

ping

- ping: Tests a Network Connection page 311

## 109.3 BASIC NETWORK TROUBLESHOOTING

Manually and automatically configure network interfaces and routing tables to include adding, starting, stopping, restarting, deleting, or reconfiguring network interfaces

Change, view, or configure the routing table and correct an improperly set default route manually

Debug problems associated with the network configuration

**Partial List of Used Files, Terms, and Utilities**

ifconfig

ifup

ifdown

route

host

- host and dig: Query Internet Nameservers page 313

hostname

- hostname: Displays the System Name page 227
- hostnamectl page 227
- Hostnames page 306
- **/etc/hostname** page 485

dig

- host and dig: Query Internet Nameservers page 313
- dig page 901
- dig page 902

netstat

ping

- ping: Tests a Network Connection page 311

traceroute

- traceroute: Traces a Route over the Internet page 312

## 109.4 CONFIGURE CLIENT SIDE DNS

*See Chapter 25: DNS/BIND: Tracking Domain Names and Addresses* page 891

Demonstrate the use of DNS on the local system

- JumpStart: Setting Up a DNS Cache page 906

Modify the order in which name resolution is done

- Resolver page 894

**Partial List of Used Files, Terms, and Utilities**

**/etc/hosts**

- Hostnames page 306
- **/etc/hosts** page 485

**/etc/resolv.conf**

- **/etc/resolv.conf** page 488

**/etc/nsswitch.conf**

- **nsswitch.conf**: Which Service to Look at First page 468

# 110 SECURITY

## 110.1 PERFORM SECURITY ADMINISTRATION TASKS

Audit a system to find files with the suid/sgid bit set

- Listing setuid files page 614
- Listing setgid files page 614

Set or change user passwords and password aging information

- User Accounts: Changing Your Account Type and Password (GUI) page 118
- passwd: Changing Your Password (CLI) page 144
- Modifying a User page 566
- chage page 567

Being able to use nmap and netstat to discover open ports on a system

Set up limits on user logins, processes, and memory usage

Basic sudo configuration and usage

- Using sudo to Gain **root** Privileges page 602

**Partial List of Used Files, Terms, and Utilities**

find

- find: Finds Files Based on Criteria page 237

passwd
- User Accounts: Changing Your Account Type and Password (GUI) page 118
- passwd: Changing Your Password (CLI) page 144

lsof
- lsof: Finds Open Files page 589

nmap

chage
- chage page 567

netstat

sudo
- Using sudo to Gain **root** Privileges page 602

**/etc/sudoers**
- sudoers: Configuring sudo page 607

su
- Using su to Gain **root** Privileges page 600

usermod
- usermod: Modifies a User Account page 567

ulimit

## 110.2 SET UP HOST SECURITY

Awareness of shadow passwords and how they work
- **/etc/shadow** page 489

Turn off network services not in use
- Server/Service (Daemon) Jobs page 430
- service page 430
- sysv-rc-conf: Configures Services page 436

Understand the role of TCP wrappers
- TCP Wrappers page 616

**Partial List of Used Files, Terms, and Utilities**

**/etc/nologin**
- Going to Recovery (Single-User) Mode page 442

**/etc/passwd**
- **/etc/passwd** page 486

**/etc/shadow**
- **/etc/shadow** page 489

**/etc/xinetd.d/*** [deprecated]
**/etc/xinetd.conf** [deprecated]
**/etc/inetd.d/*** [deprecated]
**/etc/inetd.conf** [deprecated]
**/etc/inittab**
- **/etc/inittab** page 486

**/etc/init.d/***
- SysVinit (**rc**) Scripts: Start and Stop System Services page 435

**/etc/hosts.allow**
- **hosts.allow** and **hosts.deny** page 616

**/etc/hosts.deny**
- **hosts.allow** and **hosts.deny** page 616

## 110.3 SECURING DATA WITH ENCRYPTION

Perform basic OpenSSH 2 client configuration and usage
- Configuring OpenSSH Clients page 717
- Running the ssh, scp, and sftp OpenSSH Clients page 716

Understand the role of OpenSSH 2 server host keys
- OpenSSH page 631
- Authorized Keys: Automatic Login page 728

Perform basic GnuPG configuration and usage
- Tutorial: Using GPG to Secure a File page 641

Understand SSH port tunnels (including X11 tunnels)
- Tunneling/Port Forwarding page 735

**Partial List of Used Files, Terms, and Utilities**
ssh
- ssh: Logs in or Executes Commands on a Remote System page 720

ssh-keygen
- ssh-keygen page 729

ssh-agent
- ssh-agent: Holds Your Private Keys page 731

ssh-add
- ssh-add page 732

**~/.ssh/id_rsa** and **id_rsa.pub**
- **id_xxx id_xxx.pub** page 716

**~/.ssh/id_dsa** and **id_dsa.pub**
- **id_xxx id_xxx.pub** page 716

**/etc/ssh/ssh_host_rsa_key** and **ssh_host_rsa_key.pub**
- **ssh_host_xxx_key ssh_host_xxx_key.pub** page 715

**/etc/ssh/ssh_host_dsa_key** and **ssh_host_dsa_key.pub**
- **ssh_host_xxx_key ssh_host_xxx_key.pub** page 715

**~/.ssh/authorized_keys**
- **authorized_keys** page 715

**/etc/ssh_known_hosts**
- **ssh_known_hosts** page 719

gpg
- GPG (GNU Privacy Guard) page 641
- **Tutorial: Using GPG to Secure a File** page 641

**~/.gnupg/***
- **~/.gnupg** page 641