Introduction to Command-Line Administration

2009-05-29

Contents

About This Guide

This guide provides a starting point for administering Mac OS X Server using command-line tools.

*Introduction to Scripting and Command-Line Administration*is a supplement to the information in the other advanced administration guides. It provides information useful to building workflows and remote administration practices beyond the use of Server Admin and Workgroup Manager. The information in this guide is not specific to any particular technology, but relevant to many server technologies.

What's in This Guide

This guide includes the following chapters:

- Chapter 1, "Introduction to the Command-Line Environment" provides an overview of the command-line environment in Mac OS X Server for administrators that are new to the command line in general or are coming from the command line on other platforms.
- Chapter 2, "The Interactive Shell" discusses the basics of how shells work and specific nuances of the shells in Mac OS X Server.
- Chapter 3, "Connecting to Remote Computers" provides information on various ways to access remote computers.
- Chapter 4, "Scripting the Command Line" provides an overview of shell scripting, especially in Mac OS X Server.
- Chapter 5, "Common Command-Line Tasks" provides examples of frequently used command-line task.
- Chapter 6, "Accessing Apple Hardware from the Command Line" provides information on accessing hardware-specific attributes of Macs from the command line.
- Appendix A, "Command-Line Tools Specific to Mac OS X" provides a list of the command-line tools that are unique to Mac OS X and Mac OS X Server.

Note: Because Apple periodically releases new versions and updates to its software, images shown in this book may be different from what you see on your screen.

Using Onscreen Help

You can get task instructions onscreen in Help Viewer while you're managing Snow Leopard Server. You can view help on a server or an administrator computer. (An administrator computer is a Mac OS X computer with Snow Leopard Server administration software installed on it.)

To get the most recent onscreen help for Mac OS X Leopard Server:

- Open Server Admin or Workgroup Manager and then:
  - Use the Help menu to search for a task you want to perform.
  - Choose Help > Server Admin Help or Help > Workgroup Manager Help to browse and search the help topics.

  The onscreen help contains instructions taken from *Server Administration* and other advanced administration guides described later.

To see the most recent server help topics:

- Make sure the server or administrator computer is connected to the Internet while you're getting help.

  Help Viewer automatically retrieves and caches the most recent server help topics from the Internet. When not connected to the Internet, Help Viewer displays cached help topics.

Document Road Map

Snow Leopard has a suite of guides which can cover management of individual services. Each service may be dependent on other services for maximum utility. The road map below shows some related documentation that you may need to fully configure your desired service to your specifications. You can get these guides in PDF format from the Mac OS X Server documentation website:

www.apple.com/server/resources/

Viewing PDF Guides Onscreen

While reading the PDF version of a guide onscreen:

- Show bookmarks to see the guide's outline, and click a bookmark to jump to the corresponding section.
- Search for a word or phrase to see a list of places where it appears in the document. Click a listed place to see the page where it occurs.
- Click a cross-reference to jump to the referenced section. Click a web link to visit the website in your browser.

Printing PDF Guides

If you want to print a guide, you can take these steps to save paper and ink:

- Save ink or toner by not printing the cover page.
- Save color ink on a color printer by looking in the panes of the Print dialog for an option to print in grays or black and white.
- Reduce the bulk of the printed document and save paper by printing more than one page per sheet of paper. In the Print dialog, change Scale to 115% (155% for *Getting Started*). Then choose Layout from the untitled pop-up menu. If your printer supports two-sided (duplex) printing, select one of the Two-Sided options. Otherwise, choose 2 from the Pages per Sheet pop-up menu, and optionally choose Single Hairline from the Border menu. (If you're using Mac OS X v10.4 or earlier, the Scale setting is in the Page Setup dialog and the Layout settings are in the Print dialog.)

You may want to enlarge the printed pages even if you don't print double sided, because the PDF page size is smaller than standard printer paper. In the Print dialog or Page Setup dialog, try changing Scale to 115% (155% for *Getting Started,* which has CD-size pages).

Getting Documentation Updates

Periodically, Apple posts revised help pages and new editions of guides. Some revised help pages update the latest editions of the guides.

- To view new onscreen help topics for a server application, make sure your server or administrator computer is connected to the Internet and click "Latest help topics" or "Staying current" in the main help page for the application.
- To download the latest guides in PDF format, go to the Mac OS X Server documentation website:

  www.apple.com/server/resources/
- An RSS feed listing the latest updates to Mac OS X Server documentation and onscreen help is available. To view the feed use an RSS reader application, such as Safari or Mail:

  feed://helposx.apple.com/rss/snowleopard/serverdocupdates.xml

Getting Additional Information

For more information, consult these resources:

- *Read Me documents*—get important updates and special information. Look for them on the server discs.
- *Mac OS X Server website* (www.apple.com/server/macosx/)—enter the gateway to extensive product and technology information.

- *Mac OS X Server Support website* (www.apple.com/support/macosxserver/)—access hundreds of articles from Apple's support organization.
- *Apple Discussions website* (discussions.apple.com/)—share questions, knowledge, and advice with other administrators.
- *Apple Mailing Lists website* (www.lists.apple.com/)—subscribe to mailing lists so you can communicate with other administrators using email.
- *Apple Training and Certification website* (www.apple.com/training/)—hone your server administration skills with instructor-led or self-paced training, and differentiate yourself with certification.

1 Introduction to the Command-Line Environment

Use this chapter to determine when to use command-line tools and to understand the fundamentals of how to use them.

A command-line interface (CLI) is an alternative to graphical applications for you to manipulate your computer. Mac OS X Server provides graphical applications, primarily Server Admin and Workgroup Manager, to address most common administration tasks. There are situations though, where using the command-line interface is appropriate. These situations might include:

- Configuring advanced options that aren't supported by the graphical tools.
- Remote configuration from a computer that doesn't have the Server Admin tools installed. This could include Windows, Linux, or other UNIX-based operating systems.
- Tasks that are repetitive or that need to be run at certain pre-defined times.
- Editing text files, usually by modifying advanced configuration settings and preferences.

The primary tool for accessing the CLI in Mac OS X is the Terminal application. There are other ways to access the CLI, as discussed in "Accessing the Shell" and Chapter 3, "Connecting to Remote Computers."

Each window in Terminal contains an execution context, called a *shell,* which is separate from all other execution contexts. The shell is an interactive programming language interpreter, with a specialized syntax for executing commands and writing structured programs (shell scripts). Different shells have slightly different capabilities and programming syntax. Although you can use any shell, the examples in this book use `bash`, the boot-time shell for Mac OS X and the default user shell.

The Command Line Environment

UNIX

Mac OS X and Mac OS X Server are built on the foundation of the UNIX operating system. UNIX and UNIX-based operating systems include operating systems such as BSD, GNU/Linux, AIX, and Solaris. The shared heritage of these operating systems means that many programs are compatible with minimal changes within this larger family. Programs in UNIX operating systems typically run without any sort of user interaction and have no obvious user interface other than the command line. Programs that don't have a graphical interface are often referred to as utilities. Every utility, and even high-level applications like iPhoto or Finder, are made up of multiple processes. Processes are individual programs that work together to make up the larger programs we are more familiar with, such as graphical applications and command-line utilities. When working with an operating system from the command line, you are usually launching or monitoring various processes, sometimes combining multiple processes together as steps in solving a larger problem. UNIX is essentially an amalgamation of many small tools brought together in various ways to make larger tools.

The unique underpinnings of each brand of UNIX are what distinguishes them from each other. To aid in building programs and utilities that work across multiple flavors of UNIX, there are some standard specifications proscribed by various regulating bodies. One such specification is The Open Group's "Single UNIX Specification." Mac OS X version 10.5 and later conform to version 3 of this specification. This implies conformance to the SUSv3 and POSIX 1003.1 specifications for the C API, Shell Utilities, and Threads. This means that code built to be compliant with the UNIX-03 specification will work not only on Mac OS X Server, but on any other compliant systems.

For more information about the The Single UNIX Specification, Version 3, see http://www.unix.org/version3/.

The Shell

In UNIX-based operating systems, the shell is the fundamental user interface into the system. The shell is an environment that presents a simple textual prompt to the users and accepts input from the users via the keyboard. In Mac OS X, the shell is most readily accessed through Terminal but as discussed in "Accessing the Shell," there are other options as well. The shell can be invoked interactively or via a text file with commands to the shell given in a standard format, or a shell script. There are several shells available in Mac OS X, each with its own strengths and capabilities. Shells included in Mac OS X include `bash`, `csh`, `ksh`, `sh`, `tcsh`, and `zsh`.

For information about these shells, see their man pages.

Accessing the Shell

To enter shell commands or run server command-line tools, you need access to the UNIX shell prompt on the local server or on a remote server.

Local Access

There are multiple ways to access the shell on the machine you are typing on. Under normal circumstances use Terminal, but for advanced troubleshooting or configuration, you may want to use a different mode to access the command line.

Logging In from Terminal

To open Terminal, click the Terminal icon in the dock or double-click the application icon in the Finder (in /Applications/Utilities/).

Terminal presents a prompt when it is ready to accept a command. Each window in Terminal represents another instance of a shell process. The prompt you see depends on your Terminal and shell preferences, but it often includes the name of the host you're logged in to, your current working folder, your user name, and a prompt symbol.

For example, if you're using the default `bash` shell, the prompt appears as:

```
server1:~ mariah$
```

indicates that you are logged in to a computer named *server1* as the user named *mariah,* and your current folder is mariah's home folder (~).

Logging In from the Console

You can log into a command-line version of Mac OS X without running the window manager from the login window. This mode is more advanced than "Single-User Mode" in that the entire system is running.

To log in without the Window Manager

1. In the Accounts pane of System Preferences, select Login Options.
2. Change or confirm that the settings for "Display login window as:" is set to "Name and password."
3. Log out any logged in users.
4. In the login window, type ">console" and press Return. Don't enter a password.

   You'll be prompted to login with the username and password of a user on the system.

   Logging in to the console at this level can help you troubleshoot graphics-related issues or events that are triggered by users logging in to the system through the GUI.

Single-User Mode

To debug a problem with the machine you can restart the computer and hold down Command-S as the computer boots. The computer will boot up verbosely from the command line to a certain point, and will not continue booting without your direct interaction. The window server won't be running, and many services won't be started. Onscreen instructions will guide you through mounting and verifying the attached volumes. This is a useful way to boot if you want to troubleshoot hardware-related issues or determine what is happening in software before higher-level processes and applications are running. At this point very few processes are running.

The following important processes and services are not yet running if you boot into single-user mode:

- Directory Services
- Kerberos
- `syslogd`
- mDNSResponder
- `securityd` (and many related security processes)
- Spotlight
- Any other server services (such as Mail Server, Web Server, or Wiki Server) you may have configured

X11

X11 is a window manager traditionally used in UNIX-based operating systems. Although Mac OS X Server is a UNIX operating system, it does not use X Windows as its window manager. X11 is available to provide compatibility with other UNIX-based operating systems. All normal Mac OS X Server tasks will be performed using tools that do not rely on X11. To connect to the X11 server remotely see "X11."

Serial Console

Xserve hardware includes a physical 9-pin serial port. You can connect a physical terminal or use terminal emulation software connected via a serial-to-USB cable to access the Xserve. No other Apple hardware includes a physical serial port.

Closing the Shell

To quit a particular shell session, enter the command `exit`. This ensures that any commands the shell is actively running are closed. If there is anything still in progress, the shell warns you.

Remote Access

Various ways of accessing the command-line interface on remote computers are using are discussed in Chapter 3, "Connecting to Remote Computers."

Executing Commands and Running Tools

To execute a command in the shell, enter the complete pathname of the tool's executable file, followed by arguments, and then press Return.

If a command is located in one of the shell's known folders, you can omit path information and enter just the command name.

The list of known folders is stored in the shell's PATH environment variable and includes the folders containing most command-line tools.

For example, to run the `ls` command in the current user's home folder, you could enter the following at the command line and press Return:

```
host:~ mariah$ ls
```

The shell looks through the list of folders in the PATH variable until it finds a program named `ls`; in this case, it finds `ls` in `/bin`, and runs `/bin/ls`.

To run a command in the current user's home folder, you would precede it with the folder specifier. For example, to run MyCommandLineProg, you would use the following:

```
host:~ mariah$ ~/MyCommandLineProg
```

To launch an application, you can use the `open` command:

```
open -a MyProg.app
```

When entering commands, if you get the message `command not found`, check your spelling. Here's an example:

```
server:/ mariah$ opne -a TextEdit.app
-bash: opne: command not found
```

If this error recurs, the command you're trying to run might not be in your default search path. You can add the path before the command name;

```
server:/ mariah$ sudo /System/Library/ServerSetup/serversetup
     -getHostname
server.example.com
```

or change your working folder to the folder that contains the tool:

```
server:/ mariah$ cd /System/Library/ServerSetup
server:/System/Library/ServerSetup mariah$ sudo ./serversetup
     -getHostname
server.example.com
```

or

```
server:/System/Library/ServerSetup mariah$ cd /
server:/ mariah$ PATH="$PATH:/System/Library/ServerSetup"
server:/ mariah$ sudo serversetup -getHostname
server.example.com
```

Terminating Commands

To terminate the currently running command, press Control-C. This keyboard shortcut sends an abort signal to the command. In most cases this causes the command to terminate, although commands can install signal handlers to trap this signal and respond differently.

Specifying Files and Folders

Most commands operate on files and folders, the locations of which are identified by paths. The folder names that make up a path are separated by slash characters. For example, the path to the Terminal application is /Applications/Utilities/Terminal.app.

Standard shortcuts used to represent specific folders are shown in the following table. Because they are relative to the current folder, these shortcuts eliminate the need to enter full paths in many situations.

Path string	Description
.	A single period represents the current folder. This value is often used as a shortcut to eliminate the need to enter a full path. For example, the string "./Test.c" represents the Test.c file in the current folder.
..	Two periods represent the parent folder of the current folder. This string is used for navigating up one level from the current folder through the folder hierarchy. For example, the string "../Test" represents a sibling folder (named Test) of the current folder.
~[username]	The tilde character represents the home folder of the user logged in. For example, to specify the Documents folder of the current user, you would specify ~/Documents. To specify another user's Document folder you would use their short name preceded by the tilde (~) character. For example, ~jsmith/Documents. In Mac OS X, this folder resides in the local /Users folder or on a network server. (For a list of all the shortnames on your system, type `dscl . -list /Users`. Note that most of these users are not traditional user accounts with home directories. You should though be able to find the short name of known users on the computer.)

File and folder names traditionally can include letters, numbers, a period, or the underscore character. Avoid most other characters, including space characters. Although some Mac OS X file systems permit the use of these other characters, including spaces, you might need to add single or double quotation marks around pathnames that contain them.

For individual characters, you can also "escape" the character—that is, put a backslash character immediately before the character in your string. For example, the pathname My Disk is "My Disk" or My\ Disk.

Commands Requiring Root or Administrator Privileges

Many commands used to manage a server must be executed by an administrator user or the root user. For example, entering:

```
server:~ mariah$ shutdown
```

gives you the following error:

```
shutdown: NOT super-user
```

This is because the `shutdown` command can be run only by the root user or an administrative user with an elevated mode of privileges. To run commands in this "super user" mode of privileges, use the `sudo` command. `sudo` stands for "super user do." The following command will work (but don't run it unless you really want to restart your computer!):

```
server:~ mariah$ sudo shutdown
```

You'll be prompted for the password of the currently logged in user. Only users that you have designated as Admin users are able to execute commands with `sudo`. If you are logged in as a user who isn't an admin user, you can change "substitute users" by typing `su` *`adminUsername`* where *`adminUsername`* is the name of a user in the Admin group. After entering that user's password, a new shell will be launched from the existing shell, as that user. If a command requires it, you can use `su` to log in as the root user. Under normal circumstances you don't need to use the root user account. If you do `su` to the root user, be especially careful, as you have sufficient privileges to make changes that can cause your server to stop working

For more information about the `sudo` and `su` commands, see their man pages.

Getting Help for Command-Line Tools

Using Man Pages

Most command-line documentation comes in the form of man pages. Man pages provide reference information for shell commands, tools, and high-level concepts.

You can also access command information using the `help` command, and sometimes information is displayed if you enter the command without parameters or options.

To access a man page:

```
$ man command
```

where *command* is the topic you want to find information about. The man page contains detailed information about the command, its options, parameters, and proper use.

For help using the `man` command itself, enter:

```
$ man man
```

You can press the Space bar to go to the next page, the B key to go back a page, or the Return key to scroll the file forward one line at a time. Press the Q key to exit the man page.

You can search within the contents of a man page by pressing the / key followed by the word you are looking for. If multiple instances are found, the P and N keys let you access the previous or next instances of the term.

If you don't know the name of the particular man page, you can search the topics by entering:

```
$ man -k topic
```

where *topic* is a word that would be contained in the description of the man page you might be looking for. For example:

```
$ man -k "directory service"
```

returns references to the `dscacheutil`, `dscl`, and `whois` man pages. You can also find links to related man pages at the bottom of a given man page in the "SEE ALSO" section.

If you have the Xcode tools installed, you can also view man pages from within Xcode by selecting "Open man page…" from the Help menu. There are also several third-party graphical Mac OS X applications available for viewing man pages. You can find one by choosing Mac OS X Software from the Apple menu and then searching for "man page."

Note: Not all commands and tools have man pages. Some tools use `info` pages instead, and some have no documentation at all. For more information about `info` pages, see "Info Pages."

To access in-command help:
Enter the command followed by the `-help`, `-h`, `--help`, or `help` parameter:

```
$ hdiutil help
$ dig -h
```

```
$ diff --help
```

To view a list of options and parameters you can use with the command:
Enter the command without options or parameters:

```
$ sudo serveradmin
```

Not all techniques work for all commands, and some commands don't have onscreen help.

Info Pages

Some commands use `info` pages to display their documentation. Primarily these are software packages that come from the GNU project. `info` is a tool for reading Texinfo files from the command-line. To use an `info` pages, enter the `info` command followed by the name of the tool:

```
server:/ mariah$ info emacs
```

You can navigate to nodes with the cursor and then press Return to go to them, or type `menu` followed by the node name. The following commands provide basic navigation between `info` nodes:

Key Command	Results
n	Navigates to the next page
p	Returns to the previous page
u	Navigates up one level of nodes
l	Returns to the last node visited
q	Quits the `info` programs

The Interactive Shell

2

Using the Command Line Interactively

You can use the command-line environment in Mac OS X and Mac OS X Server interactively, where you type a command and then wait for a result or you can use the shell to compose scripts that are run without direct interaction. This chapter discusses some things to keep in mind while using the command-line environment in an interactive manner.

For more information about using a specific shell interactively, see the man page for that shell.

Redirecting Input and Output

From the command line, you can redirect input and output from a command to a file or another command.

Redirecting output lets you capture the results of running the command and store it in a file for later use. Similarly, providing an input file lets you provide a command with preset input data, instead of needing to enter that data.

You can use the following characters to redirect input and output:

Redirect	Description
>	Use the greater-than character to redirect command output to a file.
<	Use the less-than character to use the contents of a file as input to the command.
>>	Use a double greater-than to append output from a command to a file.

In addition to using file redirection, you can also redirect the output of one command to the input of another using the vertical bar character, or *pipe*. You can combine commands in this manner to implement more sophisticated versions of the same commands.

For example, the command `man bash | grep commands` passes the formatted contents of the `bash` man page to the `grep` tool, which searches those contents for lines containing the word "commands." The result is a listing of lines with the specified text, instead of the entire man page.

For more information about redirection, see the bash man page.

Using Environment Variables

Some commands require the use of environment variables for their execution. Environment variables are inherited by all commands executed in the shell's context. The shell uses environment variables to store information, such as the name of the current user, the name of the host computer, and the paths to any commands.

You can create environment variables and use them to control the behavior of your command without modifying the command itself. For example, you can use an environment variable to have your command print debug information to the console.

To set the value of an environment variable, use the appropriate shell command to associate a variable name with a value. For example, to set the variable `PATH` to the value `/bin:/sbin:/user/bin:/user/sbin:/system/Library/`, you would enter the following command in a Terminal window:

```
$ PATH=/bin:/sbin:/user/bin:/user/sbin:/system/Library/ export PATH
```

This modifies the environment variable `PATH` with the value assigned.

To view all environment variables, enter the following:

```
$ env
```

When you launch an application from a shell, the application inherits much of the shell's environment, including exported environment variables. This form of inheritance can be a useful way to configure the application dynamically. For example, your application can check for the presence (or value) of an environment variable and change its behavior accordingly.

Different shells support different semantics for exporting environment variables. For further information, see the man page for your preferred shell.

Although child processes of a shell inherit the environment of that shell, shells are separate execution contexts that do not share environment information with each other. Thus, variables you set in one Terminal window aren't set in other Terminal windows.

After you close a Terminal window, variables you set in that window are gone. If you want the value of a variable to persist across sessions and in all Terminal windows, you must set it in a shell startup script.

Another way to set environment variables in Mac OS X is with a property list file in your home folder. At login, the computer looks for the ~/.MacOSX/environment.plist file. If the file is present, the computer registers the environment variables in the property list file.

Standard Pipes

Many commands can receive text input from the user and print text to the console. They do so using *standard pipes,* which are created by the shell and passed to the command.

Standard pipes include:

- `stdin`—The standard input pipe is where command input enters a command. By default, the user enters this input from the command-line interface. You can also redirect the output from files or other commands to `stdin`.
- `stdout`—The standard output pipe is where the command output is sent. By default, command output is sent to the command line. You can also redirect the output from the command line to other commands and tools.
- `stderr`—The standard error pipe is where error messages are sent. By default, errors are displayed on the command line like standard output.

Correcting Typing Errors

You can use the Left Arrow and Right Arrow keys to correct typing errors before you press Return to execute a command.

To correct a typing error:

1. Press Left Arrow or Right Arrow to skip over parts of the command you don't want to change.
2. Press Delete to remove characters.
3. Enter regular characters to insert them.
4. Press Return to execute the command.

To ignore what you entered and start again, press Control–U.

Repeating Commands

To repeat a command, press Up Arrow until you see the command, then make modifications and press Return.

Including Paths Using Drag and Drop

To include a fully qualified filename or folder path in a command, you can drag and drop the file or folder from a Finder window into the Terminal window.

Connecting to Remote Computers 3

Learn about using the command-line on machines remotely.

If you need to run command-line tools on remote computers, there are tools to help you. This topic discusses some of the most commonly used tools and provides some tips for getting started. This chapter discusses three methods for connecting to the command-line environment of a remote computer:

- SSH
- Apple Remote Desktop
- X11

SSH

SSH lets you send secure, encrypted commands to a computer remotely, as if you were sitting at the computer. You use the `ssh` tool in Terminal to open a command-line connection to a remote computer. While the connection is open, commands you enter are performed on the remote computer.

Note: You can use any application that supports SSH to connect to a computer running Mac OS X or Mac OS X Server.

How SSH Works

SSH works by setting up encrypted tunnels using public and private keys. Here is a description of an SSH session:

- The local and remote computers exchange public keys.

  If the local computer has never encountered a given public key, SSH and your web browser prompt you whether to accept the unknown key.

- The two computers use the public keys to negotiate a session key used to encrypt subsequent session data.

- The remote computer attempts to authenticate the local computer using RSA or DSA certificates. If this isn't possible, the local computer is prompted for a standard username and password combination.
- After successful authentication, the session begins and remote shell, a secure file transfer, a remote command, or other action is begun through the encrypted tunnel.

The following are SSH tools:

`sshd` Daemon that acts as a server to all other commands

`ssh` Primary user tool, which includes a remote shell, remote command, and port-forwarding sessions

`scp` Secure copy, a tool for automated file transfers

`sftp` Secure FTP, a replacement for FTP

Generating Key Pairs for Key-Based SSH Connections

By default, SSH supports the use of password, key, and Kerberos authentication. The standard method of SSH authentication is to supply login credentials in the form of a user name and password. Identity key pair authentication enables you to log in to the server without supplying a password.

Key-based authentication is more secure than password authentication because it requires that you have the private key file and know the password that lets you access that key file. Password authentication can be compromised without a private key file. A key must be generated for each user account that will use `ssh`.

How SSH key-based authentication works:

1. A private and a public key are generated, each associated with a user name to establish that user's authenticity.
2. When you attempt to log in as that user, the user name is sent to the remote computer.
3. The remote computer looks in the user's .ssh/ folder for the user's public key.

   This folder is created when using SSH the first time.
4. A challenge is sent to the user based on his or her public key.
5. The user verifies his or her identity by using the private portion of the key pair to decode the challenge.
6. After the key is decoded, the user is logged in without a password.

   This is especially useful when automating remote scripts.

   Note: If the server uses FileVault to encrypt the home folder of the user you want to use SSH to connect as, you must be logged in on the server to use SSH. Alternatively, you can store the keys for the user in a location that is not protected by FileVault, but this is not secure.

To generate the identity key pair:

1. Enter the following command on the local computer:

```
$ ssh-keygen -t dsa
```

2. When prompted, enter a filename in the user's home folder to save the keys in; then enter a password followed by password verification (empty for no password).

   For example:

```
Generating public/private dsa key pair.
 Enter file in which to save the key (/Users/mariah/.ssh/id_dsa): frog
 Enter passphrase (empty for no passphrase):
 Enter same passphrase again:
 Your identification has been saved in frog.
 Your public key has been saved in frog.pub.
 The key fingerprint is:
 4a:5c:6e:9f:3e:35:8b:e5:c9:5a:ac:00:e6:b8:d7:96 mariahjohnson1@mac.com
```

   This creates two files. Your identification or private key is saved in one file (*frog* in our example) and your public key is saved in the other (*frog*.pub in our example).

   The key fingerprint, which is derived cryptographically from the public key value, also appears. This secures the public key, making it computationally infeasible for duplication.

3. Copy the resulting public file, which contains the local computer's public key, to the .ssh/authorized_keys file in the user's home folder on the remote computer (~/.ssh/authorized_keys).

   The next time you log in to the remote computer from the local computer, you won't need to enter a password.

   If you need to establish two-way communication between servers, repeat this process on the second computer.

   Note: If you're using an Open Directory user account and have logged in using the account, you don't need to supply a password for SSH login. On computers with Mac OS X Server, SSH uses Kerberos for single sign-on authentication with any user account that has an Open Directory password. (Kerberos must be running on the Open Directory server.) For more information, see *Open Directory Administration*.

   This process must be repeated for each user who needs to be able to open a key-based SSH session. The root user isn't excluded from this requirement. The home folder for the root user on Mac OS X Server is located at /var/root/. For information about using the command-line tools mentioned here, see their `man` pages.

Key-Based SSH with Scripting Sample

A cluster of servers is an ideal environment for using key-based SSH. The following Perl script is a trivial scripting example that shouldn't be implemented. It demonstrates connecting over an SSH tunnel to all servers defined in the variable serverList, running softwareupdate, installing available updates, and restarting the computer if necessary. The script assumes that key-based SSH has been properly set up for the root user on all servers to be updated.

```
#!/usr/bin/perl
# \@ is the escape sequence for the "@" symbol.
my @serverList = ('root\@exampleserver1.example.com',
'root\@exampleserver2.example.com');
foreach $server (@serverList) {
open SBUFF, "ssh $server -x -o batchmode=yes 'softwareupdate -i -a' |";
while(<SBUFF>) {
my $flag = 0;
chop($_);
#check for restart text in $_
my $match = "Please restart immediately";
$count = @{[$_ =~ /$match/g]};
if($count > 0) {
$flag = 1;
}
}
close SBUFF;
if($flag == 1) {
\Qssh $server -x -o batchmode=yes shutdown -r now\Q
}
}
```

Updating SSH Key Fingerprints

The first time you connect to a remote computer using SSH, the local computer prompts for permission to add the remote computer's fingerprint (or encrypted public key) to a list of known remote computers. You might see a message like this:

```
The authenticity of host "server1.example.com" can't be established.
RSA key fingerprint is a8:0d:27:63:74:f1:ad:bd:6a:e4:0d:a3:47:a8:f7.
Are you sure you want to continue connecting (yes/no)?
```

The first time you connect, you have no way of knowing whether this is the correct host key. Most people respond "yes." The host key is then inserted into the ~/.ssh/known_hosts file so it can be verified in later sessions.

Be sure this is the correct key before accepting it. If possible, provide users with the encryption key through FTP, mail, or a download from the web, so they can be sure of the identity of the server.

If you later see a warning message about a *man-in-the-middle* attack (see below) when you try to connect, it might be because the key on the remote computer no longer matches the key stored on the local computer. This can happen if you:

- Change your SSH configuration on the local or remote computer
- Perform a clean installation of the server software on the computer you are attempting to log in to using SSH
- Start up from a Mac OS X Server CD on the computer you are attempting to log in to using SSH
- Attempt to use SSH to access a computer that has the same IP address as a computer that you used SSH with on another network

To connect again, delete the entries corresponding to the remote computer (which can be stored by name and IP address) in the file ~/.ssh/known_hosts.

An SSH Man-in-the-Middle Attack

Sometimes an attacker can access your network and compromise routing information, so that packets intended for a remote computer are routed to the attacker, who then impersonates the remote computer to the local computer and the local computer to the remote computer.

Here's a typical scenario: A user connects to the remote computer using SSH. By using spoofing techniques, the attacker poses as the remote computer and receives information from the local computer. The attacker then relays the information to the intended remote computer, receives a response, and then relays the remote computer's response to the local computer.

Throughout the process, the attacker is privy to all information that goes back and forth, and can modify it.

The following message, when connecting to the remote computer using SSH, can indicate a man-in-the-middle attack.

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

Protect for this type of attack by verifying that the host key sent back is the correct host key for the computer you're trying to reach. Be watchful for the warning message, and alert your users to its meaning.

Important: Removing an entry from the known_hosts file bypasses a security mechanism that would help you avoid imposters and man-in-the-middle attacks. Before you delete its entry from the known_hosts file, be sure you understand why the key on the remote computer has changed.

Controlling Access to SSH Service

You can use Server Admin to control which users can open a command-line connection using the `ssh` tool in Terminal. Users with administrator privileges can always open a connection using SSH. The `ssh` tool uses the SSH service.

For information about controlling access to the SSH service, see *Open Directory Administration*.

Connecting to a Remote Computer Using SSH

Use the `ssh` tool to create a secure shell connection to a remote computer.

To access a remote computer using ssh:

1. Open Terminal.
2. Log in to the remote computer by entering the following command:

   ```
   $ ssh -l username server
   ```

   Replace *username* with the name of an administrator user on the remote computer. Replace *server* with the name or IP address of the remote computer. For example:

   ```
   $ ssh -l mariah 10.0.1.2
   ```

   If this is the first time you've connected to the remote computer, you're prompted to continue connecting after the remote computer's RSA fingerprint appears.
3. Enter `yes`.
4. When prompted, enter the user's password for the remote computer.

   The command prompt changes to show that you're connected to the remote computer. In the previous example, the prompt might look like this:

   ```
   10.0.1.2:~ mariah$
   ```
5. To send a command to the remote computer, enter the command.
6. To close a remote connection, enter `logout`.

   You can authenticate and send a command using a single line by appending the command to the basic `ssh` tool. For example, to delete a file you could use:

   ```
   $ ssh -l mariah server1.example.com rm /Users/mariah/Documents/report
   ```

   or

   ```
   $ ssh -l mariah@server1.example.com "rm /Users/mariah/Documents/report"
   ```

   You're prompted for the user's password.

Apple Remote Desktop

Apple Remote Desktop is a software package that's sold separately from Mac OS X Server. Among the functionality in Apple Remote Desktop is a command for sending a shell script or command to client machines. This allows you to easily distribute and automate shell scripts. For more information, see the "UNIX Shell Commands" section of Apple Remote Desktop Administrator's Guide.

X11

X11 is the traditional windowing system of UNIX systems. If you're working in an environment where you need to support X11-based applications, you can use them with Mac OS X Server, but will first need to install the X11 package. The X11 server and an application to access X windows from the Finder is available as an optional installation in the Optional Installs folder of your installation DVD. Once installed, you can access an X-based terminal by launching the X11 application in /Applications/Utilities/.

The X11 implementation is based on the X.org foundation release, and is X11R7 compatible.

When using X11, keep in mind that there is a different security model then the one embraced by default with Mac OS X Server. See the X11 Preferences Security pane and this article for more information:

Configuring and Running X11 Applications on Mac OS X

Scripting the Command Line

4

What is a Shell Script?

A shell script is a text file that contains one or more UNIX commands. Shell scripts are simple programs that you can run in order to perform commands that you might otherwise run at the command-line interactively. Shell scripts are useful because you can combine many common tasks into one file, saving you time and possible errors when running similar tasks over and over. They can also be easily automated by using tools such as `launchd` or Apple Remote Desktop.

Shell scripts begin with a code that identifies it as a shell script, the characters '#' and '!' (together called a "shebang") and a reference to the specific shell that the script should be run with. For example, the following is the first line of a shell script that would be run with `sh`:

```
#! /bin/sh
```

You can, and should, comment your shell scripts. To make a comment, start a line with the pound symbol (#):

```
# This program returns the contents of my Home folder
```

You can put blank lines in a shell script to help visually distinguish different sections of the file.

The other important aspect of a shell script is that it must be made executable. This requires using the `chmod` tool to designate to the operating system that the text file can be run as a program. To make a shell script executable:

```
chmod 755 YourScriptName.sh
```

After making the shell script executable, you can run it by entering the path to the shell script and its name as follows:

```
~/Documents/Dev/YourScriptName.sh
```

or

```
cd ~/Documents/Dev/
./YourScriptName.sh
```

For more information about using `chmod`, see its man page and "." For more information about running your shell scripts, see "Executing Commands and Running Tools."

For a deeper understanding about writing shell scripts, see the Shell Scripting Primer on the Apple Developer Connection.

Monitoring and Restarting Critical Services with `launchd`

Mac OS X includes a system for monitoring and running critical services. You may want to use this service to run various shell scripts. This system is based off of a process called `launchd`. During system startup, launchd is the first process invoked by the kernel to run and set up the computer. In Mac OS X Server, your daemon should be started by `launchd`. Other mechanisms for starting daemons and services should be treated as deprecated and subject to removal at Apple's discretion. `launchd`. `launchd` is managed on a system-level or per-user basis via the `launchctl` command. You can get an idea of the various processes run by `launchd` by looking at the following configuration files:

Folder	Usage
/System/Library/LaunchDaemons/	Apple-supplied system daemons
/System/Library/LaunchAgents/	Apple-supplied agents. Apply to all users on a per-user basis.
/Library/LaunchDaemons/	Third-party system daemons
/Library/LaunchAgents/	Third-party agents. Apply to all users on a per-user basis.
~/Library/LaunchAgents/	Third-party agents. These apply to the logged-in user only.

In earlier versions of Mac OS X, a daemon called `watchdog` monitored critical services and restarted them if they failed or quit unexpectedly after a computer restarted. The `watchdog` daemon relied on the configuration file watchdog.conf, located in /etc/. In Mac OS X Server v10.4, `watchdog` was replaced by `launchd`.

Note: Some system administrators must modify the boot process to insert a script or implement a change in the default system configuration. Use `launchd` to implement changes, and avoid modifying `rc`.

For more information about `launchd`, see the `launchd` and `launchctl` man pages. Also, see Technical Note TN2083 Technical Note TN2083: Daemons and Agents.

Scheduling Shell Scripts to Run at Specific Times

To schedule tasks to run at defined times, use either `launchd` or the `cron` tool. `cron` is a daemon that executes scheduled commands defined in crontab files.

Using `cron` to schedule a task

The `cron` tool searches the /var/cron/tabs/ folder for crontab files that are named after accounts in /etc/passwd, and loads the files into memory. The `cron` tool also searches for crontab files in the /etc/crontab/ folder, which are in a different format. `cron` then cycles every minute, examining stored crontab files and checking each command to see if it should be run in the current minute.

When commands execute, output is mailed to the owner of the crontab file or to the user named in the MAILTO environment variable in the crontab file, if one exists.

If you modify a `crontab` file, you must restart `cron`.

You use `crontab` to install, deinstall, or list the tables used to drive the `cron` daemon. Users can have their own crontab file.

To configure your crontab file, use the `crontab -e` command. This displays an empty crontab file.

Here is an example of a configured crontab file:

```
SHELL=/bin/sh
PATH=/bin:/sbin:/usr/bin:/usr/sbin
HOME=/var/log
#min hour mday month wday command
30 18 * * 1-5 diskutil repairPermissions /Volumes/MacHD
50 23 * * 0 diskutil repairVolume /Volumes/MacHD
```

- The first crontab entry repairs disk permissions for the MacHD volume at 18:30 every day, Monday through Friday:

```
30 18 * * 1-5 diskutil repairPermissions /Volumes/MacHD
```

- The second crontab entry schedules a repair volume operation to run at 23:50 every Sunday:

```
50 23 * * 0 diskutil repairVolume /Volumes/MacHD
```

Scheduling tasks with `launchd`

`launchd` can be used to schedule tasks instead of `cron`. Among the benefits it offers is that if a task is skipped because the machine is off or asleep, it will be added to the queue when the computer comes back online. To use `launchd` to schedule timer-based jobs, use the *`StartCalendarInterval`* or *`StartInterval`* key. The semenatics for using this key are similar to those discussed for `crontab` in "Using `cron` to schedule a task."

For more information about `launchd`, see the `launchd` man page.

Common Command-Line Tasks

5

If you are new to using the command-line environment, it may be helpful to understand some common scenarios where people frequently use the shell instead of or in addition to the graphical interface. This section explores some of those areas and provides some guidance on getting started using the shell in these areas.

Editing Configuration Files

A common use of the command line is to manually edit configuration files to enable functionality that is not exposed in Server Admin or Workgroup Manager. In the other Server Administration documentation, you may be admonished to modify Property Lists (plists) or other regular text files to incorporate additional functionality or enforce enhanced security settings. If you are unfamiliar with using the command line to edit text files there are a few things you should be aware of including:

- Choosing an appropriate text editor
- How to edit Property List (plist) files
- Appropriate procedures for saving text files so they can be used by the UNIX subsystem of Mac OS X

Discussions of these topics follow.

Text Editors

When instructed to edit a plain text file, you need to use a text editor. Text editors are some of the oldest programs available on any operating system and come in a wide variety from completely automatic text editors where you essentially write a recipe for what actions should be taken on text and stand back and let the computer do the work to much more interactive text editors that can edit (and save) text in a wide variety of formats. For general purpose work it is easiest to deal with one of the text editors that is included with Mac OS X. If you want to use a graphical text editor, use Text Edit (in /Applications), otherwise use one of the many command line editors provided. The three most full-featured command-line text editors included with Mac OS X are:

nanoNano is a simple to use command-line based editor. It is a replacement for the Pico editor, so references to using the Pico editor can safely use nano. If you invoke the pico editor, you will actually be running in nano. Nano is a good introduction into using a command-line based editor as it includes easy to follow on-screen help.

vimVim is a vi-compatible text editor. It has many powerful enhancements above Pico for moving around, searching, and editing documents. While basic editing is simple to learn, there is a great depth of additional functionality to explore. Most functionality is accessed by typing combinations of keystroke codes that trigger certain behavior. Vim, or the editor it is modeled after, vi, is found in most UNIX-based operating systems. If you will be doing lots of editing from the command line, it is a good editor to learn to use, but if you are only using a command line-based editor occasionally, you can get by without learning it.

EmacsLike Vim, Emacs is an extremely full-featured editor. It also is found on most UNIX-based systems. Aside from its editing prowess, Emacs is extremely customizable, with functionality available in additional modules that allow the Emacs interface to do much more than just text editing. It is relatively easy to do basic editing with, and has an incredible depth of functionality for the dedicated user to explore. It also uses keystroke codes to access its many different functional behaviors. As these require memorization to be most useful, it is most useful for people that will be using the command line very often.

If you are new to using the command-line, and don't anticipate using the command-line much for editing, nano is probably the best choice. If you anticipate learning more about and spending a lot of time in the command-line environment, it is probably worth learning some of the deeper intricacies of either Vim or Emacs. With very different design philosophies, it often means spending a bit of time with each of them to determine which you work best with. For specific information on using `nano`, `vim`, or emacs, see their respective man pages.

With any of the command-line text editors, you invoke them by typing the name of the editor, followed by a space and then the name of the file you wish to open. If you want to make a new file, you can just type the name that you want to give the file. Keep in mind that you need to designate where the file is located as described in "Specifying Files and Folders." An example of using nano to open a new file named "myFile.conf" in your Documents folder is as follows:

```
$ nano ~/Documents/myFile.conf
```

Saving Text Files for UNIX Execution

When editing text files for execution by UNIX utilities, it is important that the files get saved properly so that they can be properly used (or executed) by their calling program. Two especially important things to keep in mind are using plain text and ensuring that the privileges are correct.

Using plain text

Many graphical text editors, including Text Edit that is included with Mac OS X, save text files in a format that is more complex than most UNIX programs expect. If you are using Text Edit to modify text-basd configuration files, make sure you save them as Plain Text, not the default Rich Text Format. To change the default format of text documents in Text Edit you have two options:

- To change all documents to save in plain text, select "Plain text" in the New Document Format option in Text Edit Preferences.
- To change the formatting of text in an individual document, select "Make Plain Text" from the Format menu.

While Rich Text Format appears to be simple text in most editors, it is actually a full specification that describes formatting, colors, fonts and other information that isn't found in the plain text files that most UNIX programs expect. To get an idea of what is actually contained in a Rich Text Format document, save one in Text Edit, then open the same file in a command line text editor as described in "Info Pages."

Editing Property Lists

Many preference and configuration files in Mac OS X use Property Lists (plists) to specify the attributes, or properties, of an application or a process. A common example is the Finder's user preferences plist in the Library/Preferences folder of a user's home folder. The file is named, com.apple.Finder.plist. The default naming convention for a plist includes the distributor's reverse DNS name prepended to the application or process' name and the ".plist" extension. Property lists are binary files that can be edited using the following tools:

Property List Editor A graphical application provided as a part of the Xcode tools. You can get the Xcode tools from developer.apple.com. Property List Editor is most useful if you already understand property lists and and their conventions.

`PlistBuddyPlistBuddy` is a command-line tool for directly reading and modifying values inside of a property list without the need to convert the property list to an intermediary format.

`defaultsdefaults` is a command-line utility that you can use to edit property lists.

`plutilplutil` is a command-line utility you can use to change a property list into a format you can edit with a text editor, and then back to its binary format.

Using `PlistBuddy` to edit property lists

The `PlistBuddy` command is designed to easily read and modify values in a property list. If you know the values to set or read, you can quickly make changes with `PlistBuddy`.`PlistBuddy` works on specific property list files.

To use the `PlistBuddy` command interactively to change a the orientation of the Dock for a local user:

1. Determine the names of the appropriate property list, key, and values. In this case, the name for the Dock's property list is `com.apple.Dock.plist`. If you were editing the Dock property list for the user alecjones, the path would be:

   ```
   /Users/alecjones/Library/Preferences/com.apple.Dock.plist
   ```

2. Type in the following command to enter the `PlistBuddy`interactive mode:

   ```
   PlistBuddy /Users/alecjones/Library/Preferences/com.apple.Dock.plist
   ```

   Note that if the path to `PlistBuddy` is not in your default path, you will need to add it or explicitly call it as follows:

   ```
   /usr/libexec/PlistBuddy ~/Library/Preferences/com.apple.Dock.plist
   ```

   See "Executing Commands and Running Tools."

   If the file you are trying to edit does not exist,`PlistBuddy` will create the file in the designated location.

3. In interactive mode you can choose from many commands. To set or change the orientation of the Dock to the left side of the screen, type:

   ```
   Set :orientation left
   ```

4. Save and exit with:

   ```
   Save
   ```

   ```
   Exit
   ```

   `PlistBuddy` can also be run non-interactively. To make the same change without invoking interactive mode:

   ```
   /usr/libexec/PlistBuddy -c "Set :orientation left" ~/Library/Preferences/
   com.apple.Dock.plist
   ```

   Note: Both examples above assume the *`orientation`* key already exists. This is not necessarily true for a new user in Mac OS X version 10.6. Do not assume a value exists. First, confirm it with the `Print` command. Otherwise, you need to use the `Add` command which also requires designating a type.

   There are many other options for `PlistBuddy` that are invoked in a similar manner. For information about `PlistBuddy`, see its man page.

Using the `defaults` Command to Edit Property Lists

The `defaults` command is a powerful tool with functionality beyond simple editing of property lists. When you know the specific key and value in a property list that you need to change, it is very efficient.

To use the defaults command to change a property list setting:

1. Determine the names of the appropriate property list, key, and values. In this case, the name for the Dock's property list is `com.apple.Dock.plist`. When invoking the `defaults` command, leave off the `.plist` extension.
2. Using the values you have determined or been given, type them in following the `defaults` command:

```
defaults write com.apple.dock orientation left
```

3. In most cases you'll need to restart the process or application. A simple way to do this is to use Activity Monitor to select the appropriate process, and choose "Quit Process." For the example above, you would choose the process named "Dock."

For information about `defaults`, see its man page.

Using plutil and a text editor to edit property lists

In Mac OS X v10.6, plists files are stored in a binary format by default. If you want to edit them with a text editor, you must first convert them to plain text. To convert a plist to plain text, use the `plutil` command:

```
plutil -convert xml1 com.apple.dock.plist
```

This will result in an XML text file in which you can make the appropriate changes. When you're done, you should convert the file back to the binary format with:

```
plutil -convert binary1 com.apple.dock.plist
```

Before making any changes to plist files using `plutil`, it is a good idea to make a backup of the files. You can use Command-D in the Finder, or use the `cp` command in the shell:

```
cp com.apple.finder.plist com.apple.dock.plist.bak
```

For information about Property Lists, see the `plist` man page. For the basics of command-line tool usage, see Chapter 1, "Introduction to the Command-Line Environment."

Moving and Copying Files

You can move or copy files locally and remotely using the `mv`, `cp`, and `scp` commands.

Moving files and folders locally

To move files or folders from one location to another on the same computer, use the `mv` command. The move command deletes the file from its old location and replaces it in the new location.

To move a file from your Downloads folder into a Work folder in your Documents folder:

```
mv ~/Downloads/MyFile.txt ~/Documents/Work/MyFile.txt
```

You could also change the name of the file as it's moved:

```
mv ~/Downloads/MyFile.txt ~/Documents/Work/NewFileName.txt
```

For more information about the `mv` command, see its man page.

Copying a file or folder locally

To make a copy, of a file use the `cp` command.

To copy a folder named "Expenses" in your Documents folder to another volume named "Data":

```
cp ~/Documents/Expenses /Volumes/Data/Expenses
```

You can also rename the folder on the move:

```
cp ~/Documents/Expenses /Volumes/Data/Current_Expenses
```

For more information about the `cp` command, see its man page.

Copying a file or folder remotely

To copy a file or folder from or to a remote computer, use the `scp` command. `scp` uses the same underlying protocols as `ssh`. For more information about using certificates with secure connections, see "Controlling Access to SSH Service" on page 22.

To copy a compressed file from your home folder to the *`ladmin`* user's home folder on a remote server:

```
scp -E ~/ImportantPapers.tgz ladmin@remoteserver.com:/Users/ladmin/
     Desktop/ImportantPapers.tgz
```

You will be prompted for the *`ladmin`* user's password.

The '-E' flag is unique to Mac OS X. It preserves extended attributes, resource forks, and ACL information.

For more information about the `scp` command, see its man page.

Compressing and Expanding File Archives

Mac OS X and Mac OS X use the GNU `tar` utility to compress and expand files and folders. When sending folders and multiple files between computers, it can be helpful to compress them into a single archive. This both saves space in the transfer but also means that you are transferring just one object instead of many. This can make it easier to resume should the task be suspended for some reason. The `tar` utility has many options, but for a basic compression of a folder named "LotsOfFiles" you could simply enter:

```
tar -czf LotsOfFiles.tgz LotsOfFiles
```

If it's a large folder, you may want to monitor the process by adding the 'v' flag:

```
tar -czvf LotsOfFiles.tgz LotsOfFiles
```

To open an archive, use the 'x' flag. Again, the 'v' flag is useful to see what is going on:

```
tar -xzvf LotsOfFiles.tgz
```

The 'z' flag indicates that the archive is being compressed, as well as being combined into one file. Usually you'll use this option, but you aren't required to. The traditional file extension for a that compressed archive is `.tgz`, although you might also see files ending in `.tar.gz`. If the archive is uncompressed, it usually just ends in `.tar`.

Files built with `tar` can be opened in Finder by double-clicking them. Also, if you use the File > Compress menu item in Finder to compress a folder or file, the resultant file can be opened using `tar` from the command line.

For more information about the `tar` command, see its man page.

Viewing File Contents

If you want to look at the contents of a text-based configuration file, you can use `cat` or `less`. Generally you will use `less`. To use `less`, type the command name followed by the name of the file you wish to view. The first page of text will fill the window. To view the next page, press the Space bar.

`less` also lets you search within a file. Type '/' followed by the phrase you are looking for. If the phrase has spaces in it, precede each space with '\' as follows:

```
/I\ read\ the\ other\ day
```

The following table lists some other useful keys for navigating in the output from `less`.

Key Command	Action
J or Down Arrow	Scrolls down a line
K or Up Arrow	Scrolls up a line
N	Goes to the next occurrence of a search term
P	Goes to the previous occurrence of a search term
Q	Quits `less`

For more information about the `less` command, see its man page.

Searching for Text in a File

To locate a string within a file, use the `grep` tool. The `grep` tool searches the named input files for lines containing a match to the given pattern. By default, `grep` prints the matching lines.

To search for a unique string in a file:

```
$ grep search_string filename
```

Replace *search_string* with the the string to search for and *filename* with the name of the file whose contents you want to search.

Backing Up and Restoring

While Time Machine is extremely useful for user backups, server administrators might have different needs for backing up. Mac OS X Server provides several command-line tools for data backup and restoration:

- `rsync`. Use this command to keep a backup copy of your data in sync with the original. `rsync` copies only the files that have changed.
- `ditto`. Use this command to perform full backups.
- `asr`. Use this command to back up and restore an entire volume.

For more information about these commands, see their `man` pages.

Note: You can use the `launchctl` command to automate data backup using these.

Accessing Apple Hardware from the Command Line

6

Learn how to access hardware level controls like restarting, shutting down, powering up, and selecting boot options from the command line.

This chapter introduces commands for shutting down or restarting a local or remote computer. Computers must be shut down or restarted, whether locally or remotely, when installing tools or making computer repairs.

Restarting a Computer

To restart a computer at a specific time, use the `reboot` or `shutdown -r` command. For more information, see their man pages.

To restart the local computer:

```
$ shutdown -r now
```

To restart a remote computer immediately:

```
$ ssh -l root computer shutdown -r now
```

To restart a remote computer at a specific time:

```
$ ssh -l root computer shutdown -r hhmm
```

Parameter	Description
computer	The IP address or DNS name of the computer
hhmm	The hour and minute when the computer restarts

Automatic Restart

You can also use the `systemsetup` tool to set up the computer to start up after a power failure or system freeze by specifying a number of seconds:

```
systemsetup -setwaitforstartupafterpowerfailure seconds
```

Parameter	Description
seconds	The number of seconds after which the computer will start up after a power failure. This value must be a multiple of 30.

Changing a Remote Computer's Startup Disk

You can change a remote computer's startup disk using SSH.

To determine available startup volumes:

Log in to the remote computer using SSH and enter:

```
systemsetup -liststartupdisks
```

To change the startup disk:

Log in to the remote computer using SSH, and enter:

```
CodeLinesystemsetup -setstartupdisks /Volumes/SnowLeopardServerHD/System/
     Library/CoreServices
```

For information about using SSH to log in to a remote computer, see "SSH" on page 23.

Shutting Down a Computer

To shut down a computer at a specific time, use the `shutdown` tool. For more information, see the `shutdown` man page.

To shut down a remote computer immediately:

```
$ ssh -l root computer shutdown -h now
```

To shut down the local computer in 30 minutes:

```
$ shutdown -h +30
```

Parameter	Description
computer	The IP address or DNS name of the computer

Shutting Down While Leaving the Computer On and Powered

To support UPS restart after power failure, the `shutdown` tool provides the `-u` option. This option halts system shutdown before the `shutdown` tool instructs the power manager to turn off the power supply.

The `-u` option keeps the system halted and waits for 5 minutes before removing power so an external UPS can forcibly remove power.

Using the `-u` option simulates a dirty shutdown, which allows a later automatic power on. The operating system uses the `-u` option with supported UPS devices in emergency shutdowns.

Manipulating Open Firmware NVRAM Variables

To manipulate Open Firmware NVRAM variables, use the `nvram` tool. If you change a value with `nvram`, the value is saved only if the computer cleanly restarts or shuts down.

To view NVRAM variables:

```
$ nvram -p
```

For more information, see the `nvram` man page.

Remotely Controlling the Xserve Front Panel

You can use the `ipmitool` command to remotely control the front panel of an Xserve.

To display the list of supported virtual front panel commands:

```
$ ipmitool chassis bootdev
bootdev <device> [clear-cmos=yes|no]
 none : Do not change boot device order
 pxe : Force PXE boot (LOM: Force boot NetBoot server)
 disk : Force boot from default Hard-drive
 safe : Force boot from default Hard-drive, request Safe Mode (LOM: Not
     used)
 diag : Force boot from Diagnostic Partition (LOM: Force boot diagnostic
     mode from NetBoot server)
 cdrom : Force boot from CD/DVD
 bios : Force boot into BIOS Setup (LOM: Not used)
Lights-out Management additional options
 nvram : Force reset of NVRAM
 tdm : Force boot into Target Disk Mode
 other : Skip current startup disk selection, and boot from other
```

Mac OS X Server v10.6 supports the following commands: `none`, `pxe`, `disk`, `diag`, `cdrom`, `nvram`, `tdm`, and `other`.

For example, entering the following command and then restarting an Xserve system starts the system in Target Disk Mode:

```
$ ipmitool chassis bootdev tdm
```

After the system starts, the `ipmitool` command reverts to the default setting (none). Restarting the Xserve system without running the `ipmitool` command does not change the boot device order.

For more information about `ipmitool`, see its man page.

Command-Line Tools Specific to Mac OS X

The following command line tools are unique to Mac OS X or different enough from implementations on other UNIX platforms, that it may be helpful to note them.

Please see their man pages for more information.

An online version of all of the man pages in Mac OS X and Mac OS X server is available here: http://developer.apple.com/documentation/Darwin/Reference/ManPages/

Section 1 Man Pages
Man pages in section 1 refer to commands for general tools and utilities that command-line user would use.

See the `intro(1)` man page for more information on this section.

`afconvert(1)` Audio File Convert

`afinfo(1)` Audio File Info

`afplay(1)` Audio File Play

`AppleFileServer(1)` Apple File Protocol server

`applesingle(1)` Encode and decode files

`arch(1)` Print architecture type or run selected architecture of a universal binary

`authopen(1)` Open file with authorization

`automator(1)` Runs Automator workflow

`auval(1)` AudioUnit validation

`auvaltool(1)` AudioUnit validation

`compileHelp(1)` Command-line utility to merge contextual help rtf snippets into one resource

`configureLocalKDC(1)` Generate a LocalKDC

`CreateGroupFolder(1)` Provides several options for creating and populating group directories

`createhomedir(1)` Create and populate home directories on the local computer

`defaults(1)` Access the Mac OS X user defaults system

`ditto(1)` Copy directory hierarchies, create and extract archives

`dns-sd(1)` Multicast DNS (mDNS) & DNS Service Discovery (DNS-SD) Test Tool

`drutil(1)` Interact with CD/DVD burners

`dscacheutil(1)` Gather information, statistics and initiate queries to the Directory Service cache

`dsconfigldap(1)` LDAP server config/binding add/remove tool

`dsimport(1)` Tool for importing records into an Open Directory source

`dsmemberutil(1)` Various operations for the membership APIs, including state dump, check memberships, UUIDs, etc

`emacs-undumped(1)` Basic emacs with no ELisp libraries loaded

`fs_usage(1)` Report system calls and page faults related to filesystem activity in real-time

`fwkdp(1)` FireWire KDP Tool

`fwkpfv(1)` FireWire kprintf viewer

`genstrings(1)` Generate string table from source code

`hdiutil(1)` Manipulate disk images (attach, verify, burn, etc)

`hiutil(1)` Manipulate disk images (attach, verify, burn, etc)

`inituser(1)` Creates a user's home directory and sets quota

`languagesetup(1)` Set the primary language

`latency(1)` Monitors scheduling and interrupt latency

`launchctl(1)` Interfaces with launchd

`locale(1)` Display locale settings

`localedef(1)` Define locale environment

`mDNS(1)` Network diagnostic tool

`migrateLocalKDC(1)` Migrates a LocalKDC

`MP3Broadcaster(1)` MP3 file playlist broadcaster

`netstat(1)` Show network status

`notificationconf(1)` Configures how aosnotifyd responds to the NSServerNotificationCenter api

`notifyutil(1)` Notification command line utility

`ocspd(1)` OCSP and CRL Daemon

`open(1)` Open files and directories

`opendirectorypdbconfig(1)`

`osacompile(1)` Compile AppleScripts and other OSA language scripts

`osadecompile(1)` Display compiled AppleScripts or other OSA language scripts

`osalang(1)` Information about installed OSA languages

`osascript(1)` Execute AppleScripts and other OSA language scripts

`passwd(1)` Modify a user's password

`pcast(1)` Podcast Producer command line tool

`pcastaction(1)` Podcast Producer action command line tool

`pcastconfig(1)` Podcast Producer server configuration command line tool

`pkgutil(1)` Query and manipulate the installer package receipt database

`pl(1)` Server stress test program run MySQL test suite ASCII property list utility Extract translatable strings from source

`PlaylistBroadcaster(1)` Hinted media playlist broadcaster

`plutil(1)` Property list utility

`pmset(1)` Manipulate power management settings

`podcast(1)` Podcast Producer command line tool

`qc2movie(1)` Quartz Composer export tool

`qtpasswd(1)` MP3 file playlist broadcaster

`sandbox-exec(1)` Execute within a sandbox

`sandbox-simplify(1)` Simplify a sandbox profile created by a trace directive

`sar(1)` System activity reporter

`sc_usage(1)` Show system call usage statistics

`security(1)` Command line interface to keychains and Security framework

`securityd(1)` Security context daemon for Authorization and cryptographic operations

`ServerBackup(1)` The main program that will perform actions supported by the Server Backup application

`setquota(1)` Sets home directory quotas on the local computer

`sips(1)` Scriptable image processing system

`syslog(1)` Apple System Log utility

`tconf(1)` TargetConfig command line tool

`textutil(1)` Text utility

`tiff2icns(1)` Converts TIFF to icns format

`tiffutil(1)` Manipulates tiff files

`update_dyld_shared_cache(1)` Updates dyld's shared cache

`uuidgen(1)` Generates new UUID strings

`vm_stat(1)` Show Mach virtual memory statistics

`wait4path(1)` Wait for given path to show up in the namespace

`xgrid(1)` Submit and monitor xgrid jobs

Section 4 Man Pages

Man pages in section 4 refer to descriptions of special files and devices.

`dummynet(4)` Traffic shaper, bandwidth manager and delay emulator

`ipfirewall(4)` IP packet filter and traffic accounting

`random(4)` , urandom random data source devices

Section 5 Man Pages

Man pages in section 5 give information about file formats and conventions.

See the `intro(5)` man page for more information on this section.

`af.plist(5)` Config file for afctl

`asl.conf(5)` Configuration file for syslogd(8) and aslmanager(8)

`auto_master(5)` Automounter master map

`autofs.conf(5)` automount(8) and automountd(8) configuration file

`bom(5)` Bill of materials

`bootparams(5)` Boot parameter database

`bootptab(5)` Internet Bootstrap Protocol server database

`compat(5)` Manipulate compatibility settings

`emond.plist(5)` Config file for emond

`fstab(5)` Static information about the filesystems

`group(5)` Format of the group permissions file

`launchd.conf(5)` launchd configuration file

`launchd.plist(5)` System wide and per-user daemon/agent configuration files

`plist(5)` System wide and per-user daemon/agent configuration files property list format

`resolver(5)` Resolver configuration file format

`types(5)` Mime type description file for cups system data types

Section 7 Man Pages

Man pages in section 7 are miscellaneous pages that do no belong in any other specific sections.

See the `intro(7)` man page for more information on this section.

`sandbox(7)` Overview of the sandbox facility

Section 8 Man Pages

Man pages in section 8 document commands that system administrators would invoke as well as daemons.

See the `intro(8)` man page for more information on this section.

`afctl(8)` Automatic host blocking

`aosnotifyd(8)` Apple Online Services notifcation daemon

`aslmanager(8)` Configuration file for syslogd(8) and aslmanager(8) Apple System Log data store file manager

`asr(8)` Apple Software Restore; copy volumes (e.g. from disk images)

`atrun(8)` Run jobs queued for later execution

`atsutil(8)` Font registration system utility

`autodiskmount(8)` Disk support tool

`autofsd(8)` Daemon to update autofs mounts on network changes

`automount(8)` Automount(8) and automountd(8) configuration file mount autofs on the appropriate mount points

`automountd(8)` Automount(8) and automountd(8) configuration file automatic mount / unmount daemon for autofs

`bless(8)` Set volume bootability and startup disk options

`blued(8)` The Mac OS X bluetooth daemon

`bootpd(8)` DHCP/BOOTP/NetBoot server

`cd9660.util(8)` ISO 9660 file system utility

`certadmin(8)` Mac OS X Server certificate administration helper tool

`chkpasswd(8)` Verifies user password against various systems

`configd(8)` System Configuration Daemon

`coreaudiod(8)` Daemon used for Core Audio related purposes

`createuserstreamingdir(8)` Create the QuickTime Streaming Server directory

`cvt_mail_data(8)` Sends events to the Event Monitor (emond)

`dirhelper(8)` Helper for special directory creation

`diskarbitrationd(8)` Disk arbitration daemon

`diskmanagementd(8)` DiskManagement.framework server

`disktool(8)` Disk support tool

`diskutil(8)` Modify, verify and repair local disks

`distnoted(8)` Distributed notification server

`dnsextd(8)` BIND Extension Daemon

`docsetbless(8)` Documentation set installation blesser

`docsetinstalld(8)` Documentation set installer daemon

`dovecotd(8)` Dovecot mail deamon

`dovecotpw(8)` Utility which can be used to easily generate passwords for a wanted scheme

`dsconfigad(8)` Retrieves/changes configuration for Directory Services Active Directory Plugin

`dseditgroup(8)` Group record manipulation tool

`dsenableroot(8)` Enables or disables the root account

`dumpemacs(8)` Utility to dump pre-loaded emacs with compiled ELisp auto-loads

`dynamic_pager(8)` External storage manager for dynamic pager

`emlog.pl(8)` Scans log messages for authentication failures and other security notices

`emond(8)` Event Monitor Daemon

`fibreconfig(8)` Tool for configuring settings for Fibre Channel controllers and targets

`firmwaresyncd(8)` Synchronize files used by the system firmware

`fontd(8)` Mac OS X system font registration manager

`fontmover(8)` Mac OS X system font mover

`fontworker(8)` Mac OS X system font registration and validation daemon

`fsck_hfs(8)` HFS file system consistency check

`gssd(8)` Generic Security Services Daemon

`hdik(8)` Lightweight in-kernel disk image mounting tool

`hfs.util(8)` HFS/HFS+ file system utility

`hostinfo(8)` Host information

`ifcstart(8)` Rebuilds international data caches

`installer(8)` System software and package installer tool

`InternetSharing(8)` Simple NAT/router configuration daemon

`ioprofilepolicyd(8)` IOProfileFamily access control daemon

`ioreg(8)` Show I/O Kit registry

`ioupsd(8)` Daemon to track UPS state

`ipconfig(8)` View and control IP configuration state

`ipfw(8)` IP firewall and traffic shaper control program

`kadmin_util(8)` Manage Kerberos ACLs

`kdcsetup(8)` Configure an Apple Open Directory KDC

`kerberosautoconfig(8)` Creates, updates & removes the edu.mit.Kerberos file

`kext_logging(8)` Verbose/logging flags for kernel extensions (kexts) in the kernel and command-line utilities

`kextcache(8)` Create kext cache files

`kextd(8)` Kernel extension server

`kextfind(8)` Find kernel extensions (kexts) based on a variety of criteria and print information

`kextlibs(8)` Find OSBundleLibraries needed by a kext

`kextload(8)` Load kernel extensions (kexts) into the kernel

`kextstat(8)` Display status of loaded kernel extensions (kexts)

`kextunload(8)` Terminate driver I/O Kit driver instances and unload kernel extensions (kexts)

`kextutil(8)` Load, diagnose problems with, and generate symbols for kernel extensions (kexts)

`krbservicesetup(8)` Configure Kerberized services on the current host

`kuncd(8)` The Kernel User Notification Center daemon

`launchd(8)` System wide and per-user daemon/agent manager

`launchproxy(8)` Inetd job emulation helper

`lsbom(8)` List contents of a bom file

`mDNSResponder(8)` Multicast and Unicast DNS daemon

`mDNSResponderHelper(8)` mDNS privilege separation helper

`mds(8)` Metadata server

`mdworker(8)` Metadata server worker

`memorytracker(8)` Tracks memory statistics and reports the information back to Apple for analysis

`mkbom(8)` Create a bill-of-materials file

`mkextunpack(8)` Extract or list the contents of a multikext (mkext) archive

`mount_cddafs(8)` Mount an Audio CD

`mount_hfs(8)` Mount an HFS/HFS+ file system

`mount_ntfs(8)` Mount an NTFS file system

`mount_url(8)` Mount a remote file system given a URL

`msdos.util(8)` DOS/Windows (FAT) file system utility

`nat_start(8)` Tool to start natd on Mac OS X Server

`nat_stop(8)` Tool to stop natd on Mac OS X Server

`natd(8)` Network Address Translation daemon

`nbdst(8)` NetBoot deferred shadow tool

`networksetup(8)` Configuration tool for network settings in System Preferences

`newfs_hfs(8)` Construct a new HFS Plus file system

`newfs_udf(8)` Construct a new UDF file system

`notifyd(8)` Cyrus notification server notification server

`ntfs.util(8)` NTFS file system utility

`ntpd-wrapper(8)` Wrapper for ntpdate/ntpd called by launchd

`path_helper(8)` Helper for constructing PATH environment variable

`pboard(8)` Pasteboard server

`pbs(8)` General helper tool

`pcastctl(8)` Podcast Producer daemons control interface

`pictd(8)` General helper tool

`PlistBuddy(8)` Read and write values to plists

`pmap_dump(8)` Print a list of all registered RPC programs

`pmap_set(8)` Set the list of registered RPC programs

`push_notify(8)`

`QuickTimeStreamingServer(8)` RTP/RTSP streaming server

`rc(8)` Deprecated

`RegisterSeRV(8)` Mac OS X serial number registration tool

`repair_packages(8)` Verify or repair filesystem permissions and flags for packages

`ReportCrash(8)` Generates crash reports

`sa1(8)` Generate a system activity daily data file

`sa2(8)` Generate a system activity daily data file

`sadc(8)` System activity data collector

`sandboxd(8)` Sandbox daemon

`scselect(8)` Select system configuration location

`scsid(8)` SCSI subsystem daemon

`scutil(8)` Manage system configuration parameters

`security_authtrampoline(8)` Trampoline used by AuthorizationExecuteWithPrivileges from Security.framework

`service(8)` Deprecated

`service_helper(8)` Helper program for enabling and disabling services

`setregion(8)` Set the disc region code for a DVD drive

`softwareupdate(8)` Software Update checks for new and updated versions of your software

`spindump(8)` Report generation for unresponsive applications helper process for spindump(8)

`spindump_symbolicator(8)` Helper process for spindump(8)

`sso_util(8)` Tool for setting up, interrogating and removing Kerberos configurations within the Apple Single Sign On environment

`StartupItemContext(8)` Execute a program in StartupItem context

`streamingadminserver.pl(8)` Web server that manages the html for QuickTimeStreamingServer web-based administration

`StreamingLoadTool(8)` Load testing tool for QuickTime Streaming Server

`SubmitDiagInfo(8)` Sends diagnostic information to Apple

`syslogd(8)` Configuration file for syslogd(8) and aslmanager(8) syslogd(8) configuration file Apple System Log server

`system_profiler(8)` Reports system hardware and software configuration

`systemsetup(8)` Configuration tool for certain machine settings in System Preferences

`SystemStarter(8)` Deprecated

`taskgated(8)` Task_for_pid access control daemon

`tokenadmin(8)` Command-line interface to smartcards and other token-based keychains

`ufs.util(8)` UFS file system utility

`upsshutdown(8)` UPS emergency low power shutdown script

`UserEventAgent(8)` High-level system event handler

`vpnd(8)` Mac OS X VPN service daemon

`vsdbutil(8)` Manipulates the volume status DB

`warmd(8)` Pre-heating daemon

`warmd_agent(8)` Pre-heating agent

`wikictl(8)` Mac OS X Server Wiki Server Control Interface

`xgridctl(8)` Xgrid Daemon Control Interface

`xssendevent(8)` Event Monitor Tools