

MACWORLD'S 50 TOP

# Mac OS X Tips and Unix Tricks

O'REILLY'S  
ESSENTIAL



# TABLE OF CONTENTS

## **50 Top Mac OS X Tips**

General System Tips **2**

Software **11**

Hardware/Compatibility **14**

## **Unix Tricks**

The Unix Shell **17**

The Unix File System **22**

Input and Output **30**

# 50 Top Mac OS X Tips

by Christopher Breen

## GENERAL SYSTEM TIPS

### 1. OS X STARTUP

You'll notice that when you boot from a volume running Mac OS 9 while holding down the option key, the Startup Manager on your iMac (slot loading), iBook, PowerBook (FireWire), or Power Mac G4 (AGP graphics) fails to reveal any OS X disks. To start up in Mac OS X from OS 9, you should instead open the System Disk control panel, select the disk you want to use, close the control panel, and reboot your Mac.

### 2. TRULY BLESSED

Mac OS 9's System Disk control panel and OS X's System Preferences application-programs that let you choose whether to boot your Mac from a volume running OS 9 or one running OS X-don't distinguish between "blessed" and "unblessed" System Folders. Instead of providing you with a list of only those System Folders that carry the Mac smiley face icon on the folder (blessed folders), all System Folders (including previous System Folders) appear as selectable options. It's not a good idea to select a System Folder other than the one that's blessed.

Fortunately, it's easy to tell the blessed from the unblessed. If you see an entry in System Disk or System Preferences that looks like *Bubba's Mac: System Folder*, you're OK. If the entry reads *Bubba's Mac: Previous System Folder*, stay away!

### 3. CLASSIC OVERLAP

While running in OS X's Classic environment, if an object such as a tool tip label (the little box that pops up to describe the purpose of a particular toolbar tool) protrudes into the OS X desktop, that object may be cut off. This clearly demonstrates that classic Mac OS and OS X are completely different beasts: When the classic as attempts to take over real estate held by OS X, OS X simply won't allow it.

What do you do about it? Place a document from another classic application behind the application you're using. For example, if you put a blank SimpleText document behind the application with the tool tips,

those tool tips should extend into SimpleText's screen real estate—real estate held by the classic OS—and display properly.

## **4. PERSONAL FILE SHARING UNDER OS X**

If you use Personal File Sharing, you may notice that when you open the File Sharing control panel in the Classic environment under OS X and try to select the Enable File Sharing Clients To Connect Over TCP/IP option, an alert box appears that informs you that you're out of luck.

Thankfully, you're only out of luck as far as the Classic environment is concerned. If you want to share files over TCP/IP, you can do it natively within OS X. Just launch the System Preferences application, click on the Sharing icon, and then click on the Start button.

## **5. OS X SLOWDOWNS**

If you're running OS X you may find that performance decreases after you log in and out. According to Apple, this is caused by using the Classic environment over a number of sessions; the solution is to restart your Mac. However, if the Mac takes a performance hit when the Classic environment is repeatedly run, it might make sense to leave the Start Up Classic On Login To This Computer option deselected in the Classic preference.

## **6. MOUNTCHECK ERRORS**

If you've run Disk First Aid (DFA) 8.5.5 on an OS X volume, you've probably been notified that MountCheck has found minor errors. Attempting to repair these errors with DFA does no good: although DFA reports success, the error message persists on subsequent diagnoses.

These error messages are, themselves, errors. Versions of DFA prior to 8.6 spit out this bogus error because they were unfamiliar with OS X's unique structure.

To find the latest version of Disk First Aid, use the Search feature on Apple's Featured Software site ([www.info.apple.com/support/downloads.html](http://www.info.apple.com/support/downloads.html)).

## **7. HANDY OS X ALIAS**

With the release of OS X, Apple bundled a bunch of worthwhile software into the iTools iDisk, the virtual hard disk that resides on Apple's servers. When Apple places a new piece of software on your iDisk, it can be found at Software: Mac OS X Software: What's New. Because navigating through an iDisk is anything but speedy, you'd be well-served to create an alias of the What's New folder and tuck it away in a convenient location on your hard drive.

The next time you want to check on the contents of this folder, just double-click on its alias.

## 8. THE GENIE SUCKS?

Mac OS X includes an effect that causes minimized windows to swoop into and out of the Dock like a genie from a bottle. Some OS X users with only



moderately fast Macs have found that this effect is not terribly smooth.

Thankfully, OS X includes two other effects: Suck, which looks a bit like the default Genie effect, and Scale, which pulls the window down into the Dock more smoothly but with less swoop.

### Use Dock preferences to change the minimize effect.

To change the effect, go to the System Preferences application and click on Dock. You can choose Scale and Genie from the Minimize Using pop-up window. If you want to tryout the Suck effect, you must enter a few commands in OS X's Terminal application (found inside the Utilities folder inside OS X's Applications folder).

After launching Terminal, type `defaults write com.apple.Dock mineffect suck`.

Then press the return key and type `exit`. Close Terminal and log out of OS X .

When you log back in, the Suck effect will be active.

To return to the Genie or Scale effect, just select it in Dock preferences.

## 9. FORCE QUIT FROM THE DOCK

You probably know that you can press `⌘-option-shift-escape` or choose Force Quit from the Apple menu to force-quit an application in OS X.

You can also force-quit an application by option-clicking on the application's icon in the Dock, and choosing Force Quit from the pop-up menu that appears.

## 10. GENERIC DOCK ICONS

If OS X's Dock displays generic folder icons when it should be showing

application icons, try this:

Drag the LSApplications, LSCLaimedTypes, and LSSchemes files to the Trash. If the icons are generic when the user "Johnny" is signed on, for example, navigate to these files from the root Mac OS X disk via Users: Johnny: Library: Preferences.

Now log out of OS X and log back in as the affected user.

## **11. LOCKED OS X FILES IN THE TRASH**

Under classic Mac OS, you can delete locked files in the Trash by holding down the option key and selecting Empty Trash from the Special menu. This trick won't work under OS X. Instead, open the Trash, click on the item you'd like to delete, press ⌘-I (Get Info), and deselect the Locked option. Once you've done so you can discard the file.

## **12. REINSTALL WITHOUT REGISTERING**

If you've had occasion to reinstall OS X, you've undoubtedly run up against the registration screen that asks for your name, address, e-mail address, and so forth. You don't want to rankle Apple by sending this information time and again, yet the setup assistant won't let you proceed without entering this personal information. What do you do?

Quit, that's what. Just press ⌘-Q. The assistant will fling forth a dialog box with a

Skip option that lets you bypass the personal interrogation and get on with the setup.

## **13. PAINT IT BLACK**

There are times when you may want to keep your LCD screen running on full-when you need your PowerBook to be completely awake, for example-but you'd like to keep all those pixels dark in order to prolong the display's life. Here's how to do it under OS X.

Create a completely black image in a graphics application and save it to OS X's Pictures folder inside your Users folder. Now launch the Screen Saver system preference, click on the Custom Slide Show option, and close the Screen Saver preference. When your screen saver next kicks in, you'll see nothing but a black screen.

## **14. DESKTOP FILES**

If you switch between OS X and OS 9 and routinely save files to the desktop, you may wonder where the files that once resided on your desktop have gone. Under OS X you can find your OS 9 desktop files in a folder called, aptly enough, Desktop Folder, at the root level of the drive on which your OS 9 System Folder lives.

When running OS 9 you'll find OS X's desktop files inside OS X Volume: Users: Chris: Desktop, using the user "Chris" as an example.

## 15. UFS LIMITATIONS

When you install OS X on a partition or volume formatted as Unix File System (UFS) rather than HFS+ (Mac as Extended), you'll face a few limitations.

To begin with, AirPort cards aren't recognized on partitions formatted this way.

Also, the Classic environment won't work the first time you attempt to launch it. If you want to make Classic function under such a formatting scheme, Apple suggests you do the following:

1. Log in to Mac OS X as an admin user.
2. Choose Go To Folder from the Go menu.
3. Type /System/Library/CoreServices
4. Click on Go.
5. Drag the item Classic Startup to the OS 9.1 volume.
6. Open the Terminal utility.
7. Type `cd/Volumes/HFSvolume/Classic/Startup.app/Contents/Resources`  
Note: In place of HFSvolume, type the exact name of your Mac OS 9.1 volume.
8. Press return.
9. Type `chmod u+s TruBlueEnvironment` and press return.
10. Type `sudo chown root TruBlueEnvironment` and press return.
11. Enter your admin user password at the prompt and press return.
12. Quit Terminal.
13. Open Classic Startup from the Mac OS 9.1 volume.
14. Click on OK when prompted to add resources.

## 16. OS X AND THE OPTION KEY

You probably know that under classic Mac OS, holding down the option key changes some Finder and application commands. For example, if you option-click on the Finder's File menu, the Close command changes to Close All.

OS X offers this feature with a slight difference: when a menu is pulled down, pressing the option key updates menu commands in real time. That's right: if you have the Finder's Windows menu pulled down and press the option key, you'll see the Minimize Window command change to Minimize All Windows and the Bring All To Front command switch to Arrange In Front.

## 17. BEHIND THE SCENES

Want to know exactly what Mac OS X is doing while it boots? Press  $\text{⌘-V}$  just after pressing the power key. Doing so launches the Mac into "verbose" mode, which trails scads of commands that only a developer could love across your screen as the Mac boots up.

## 18. SPECIAL CHARACTERS AND HELP

If your OS X volume contains the backslash (\) or trademark symbol (TM), OS X's Help Viewer will report that "The specified HTML file could not be found" when you select a help topic. The workaround is to remove these symbols from the volume's name.

## 19. KEEPING TRACK OF MEMORY

In OS 9.2 and earlier you can see how much memory your Mac has and how it is allotted by selecting About This Computer from the Apple menu. Only part of this procedure applies under OS X.

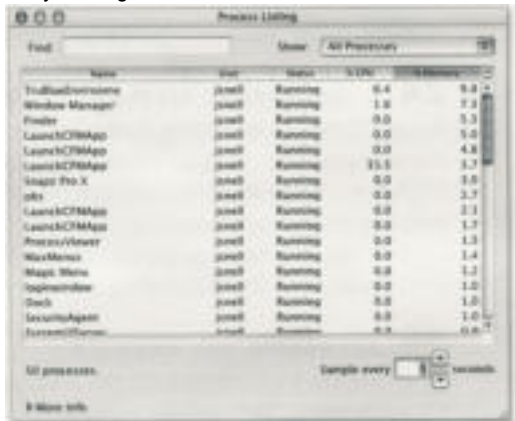
To see how much memory you have under OS X, select About This Mac from the Apple menu. To see where your RAM is going, open the Process Viewer application (found in the Utilities folder of the Applications folder). You'll see how your RAM is portioned out in the % Memory column to the right.

## 20. NEW ALERT SOUNDS

To add new alert sounds to OS X, try this:

Open the Sounds folder inside the Library folder inside your Users folder (you can quickly navigate to your Users folder by clicking on the Home button in a Finder window). Drag an AIFF sound file from another Finder window or from the Desktop into your Sounds folder. Select System Preferences from the Apple menu and click on the Sound icon. Select your new sound in the Alert Sound field.

**Note:** The sound file must bear the *.aiff* suffix (*aif* won't do). Also, if System Preferences



Name	PID	Status	CPU	% Memory
TrustedInfoSource	1000	Running	0.4	0.0
Window Manager	1001	Running	1.8	7.3
Finder	1002	Running	0.0	5.3
LaunchDaemon	1003	Running	0.0	0.0
LaunchDaemon	1004	Running	0.0	4.8
LaunchDaemon	1005	Running	35.5	3.7
Kernel Task	1006	Running	0.0	0.0
off	1007	Running	0.0	3.7
LaunchDaemon	1008	Running	0.0	3.3
ProcessViewer	1009	Running	0.0	1.7
MacOSX	1010	Running	0.0	1.3
Maple Menu	1011	Running	0.0	1.2
SystemUIServer	1012	Running	0.0	1.0
Dock	1013	Running	0.0	1.0
SecurityAgent	1014	Running	0.0	1.0
SystemUIServer	1015	Running	0.0	0.6

The ProcessViewer application shows RAM allocation.

Was open before you moved the sound file to the Sounds folder, you might have to quit and relaunch System Preferences before the file is recognized.

## **21. STARTUP TRICK**

Hold down `⌘-S` while starting up under OS X . You'll see code scroll across your screen; at some point the scrolling will stop and you'll be presented with the command prompt. If you're Unix-savvy, you can take this opportunity to log on to your Mac before Aqua loads, thus allowing you to delete troublesome files, create directories, and perform other Unix magic.

To allow the as to continue loading, simply type `exit` and press the return key.

## **22. DON'T RENAME UTILITIES**

If you rename OS X's Utilities folder, you alter the pathname to that folder and therefore interrupt any processes that require it. For example, you can't print with this folder's name changed because OS X expects Print Center to be inside a folder called Utilities. When OS X finds it isn't, the process gives up and refuses to print.

## **23. DISABLE AUTO-DIAL**

If you don't want the modem on your Mac running OS X to automatically dial your ISP when an Internet-based action is initiated, you can disable the auto-dial feature.

Here's how:

Open System Preferences, click on the Network preference, select your modem from the Configure pop-up menu, click on the ppp tab, and then click on PPP Options. Deselect the Connect Automatically When Starting TCP/IP Applications option. Click on OK and save.

## **24. UNNECESSARY DISPLAY SOFTWARE**

If you attempt to install the Apple Display Software that comes with the Apple Studio Display 17 (LCD) on a Mac running OS X 10.0.4 or later or OS 9.2, you'll see an error message that states "This software cannot be installed on this computer."

Fear not. this message appears only because this software is unnecessary with these operating systems-it's already installed with them.

## **25. ACCESSING THE CONSOLE**

To access OS X's Console from the login screen, select Log Out from OS X's Apple menu. Type console in the Name field. Don't bother to enter a password, and press Log In. In a flash, you'll see the command-line con-

sole, ready to do your bidding.

Those who have upgraded to OS X 10.1 may very well ask, "What name field? All I see on the login screen is a list of users with an icon next to each user's name. This screen carries no fields whatsoever."

And they'd be right. By default, the login screen doesn't carry name and password fields. To cause these fields to appear, open System Preferences and choose the Login preference. Click on the Login Window tab and select the Name And Password Entry Fields option in the Display Login Window area. When you next log in to OS X, you'll see the fields.

Apple knew what it was doing by leaving this option off by default. Allowing access to the name and password fields means that others can use the console trick and have ready access to the command-line interface on your Mac (though they can use the command line if they booted the Mac while holding down  $\text{⌘}$ -S, as well).

## 26. RE-CREATING THE DESKTOP ALIAS

If OS 9.X and OS X are installed on the same disk and you delete the Desktop alias (Mac OS 9), you can't get that alias back by normal means (navigating to the OS Desktop folder and creating an alias). Instead, you must open OS X's Terminal application and type `In -s /"Desktop Folder" -/Desktop/"Desktop (Mac OS 9)"` and press return.

Now when you click on the desktop, you should see the Desktop (Mac OS 9) alias.

## 27. IMAGE CAPTURE

Those who saw Steve Jobs's keynote address at July 2001's Macworld Expo know that if you plug a digital camera into a Mac running OS X 10.1, the pictures stored on the camera's media card will automatically be sucked into the Mac. (And no, you needn't first hurl the camera to a nearby subordinate to make it happen.)

What Apple didn't make particularly clear is that you don't need a direct USB-to-camera connection for this to work. You can, for example, plug a supported USB card reader into your Mac, shove a media card into the reader, and Image Capture should pull the images into the Mac.

## 28. OS X 10.1 ABOUT BOX

The OS X 10.1 About box has a hidden feature. (Now don't get your hopes up: it's not as cool as the old About The Finder Easter Egg in OS 9.2 and earlier.)

Select About This Mac from the Apple menu and click once on

the version number. You'll see the build version of OS X . Click on it again and your Mac's serial number will display (if your Mac supports serial number display-not all do).

## **29. THE SLOW-MOTION DOCK TRICK**

If you've witnessed just about any demo of OS X, you've seen the minimize-in-slow-mo trick where a window is ever-so-slowly sucked into the Dock. To perform this same trick with your copy of OS X, just shift-click on a window's Minimize button.

## **30. OS X TOOLBAR**

You probably know that to customize the OS X Toolbar, you simply press the Toolbar button (that clear button on the right corner of a window) while holding down the shift key.

But you may not know that to pull this same kind of customization trick with OS X applications (those that support OS X tool bars, natch), you hold down the ⌘ and option keys while clicking on this button.

## **31. OPEN FIRMWARE UPDATES AND OS X**

Firmware updates cannot be installed when you boot your Mac from OS X . To perform such updates, first boot from OS 9.

## **32. UNDERPRIVILEGED**

OS X's ability to share files is limited compared with earlier versions of Mac OS. Although OS X hints that you *can* share all the files on your hard drive via a file's Privileges window, you can't. You can share only the files in your user folder's Public folder.

The only way to see all the files on an OS X disk is to log on as an administrator.

(By the way, if you want to check a file's privileges, you must click on the file, press ⌘-I, and select Privileges from the resulting Get Info window's pop-up menu.)

## **33. DELETING OS X PRINT .JOBS**

It's fairly easy to stop a print job under OS 9.2 and earlier. Just double-click on the desktop printer icon, select the document you want to delete, and click on the Trash icon.

OS X doesn't support desktop printers, so this procedure won't work. To delete print jobs in OS X, you must do the following:

Click and hold on the Print Center icon in the Dock and select Show Queues in the resulting contextual menu. Click on the job you want to

delete and press the Delete button in the printer window. To delete multiple jobs, ⌘-click on the jobs you want to delete and click on this same button.

## 34. ACCESS CONTROL PANELS FROM OS X

OS X users who occasionally switch to the Classic environment may notice that you can access Classic's Apple menu only if a Classic application is running. This can be mighty annoying if you only want to launch a control panel.

To work around this problem, make an alias of Classic's Control Panels folder and place that alias in OS X's Dock. It will then open the Control Panels menu for you. For example, when you want to open Classic's Software Update control panel, just click and hold on the Control Panels folder alias in the Dock, wait for the hierarchical menu to appear, and select Software Update.

## 35. EASY CLASSIC START-UP

To easily start up the Classic environment under OS X, don't go to the Classic system preference. Instead, go to the System: Library: Core Services folder, then drag the Classic Startup application to the Dock.

When you next want to launch Classic, just click on the Classic Startup icon in the Dock. (Thanks to John Welch of TackyShirt.com for this tip.)

## 36. OS X, APPLEWORKS, AND FONT SMOOTHING

If you switch off font smoothing within AppleWorks 6 under OS X, the cursor will be displaced by a couple of characters. Unless you really like this effect, choose Preferences from the AppleWorks menu, select Text from the Topic menu, and enable the Font Smoothing option.

# SOFTWARE

## 37. RESTORE BUNDLED APPS

Apple generously bundles iMovie, iTunes, and iPhoto with new Macs. Unfortunately, these apps are packed onto Software Restore discs,



Control panels in the Dock.

which provide no way to easily install individual applications. Instead, you have to install everything-and in some cases, erase the contents of your hard drive-unless you know the following trick:

Clear 2GB of hard disk space and create a new folder. Into this folder, copy the disk-image files (these may be in a Configurations folder) from each of the Software Restore discs that came with your Mac. Launch Disk Copy and drag the first disk-image file into the Disk Copy window. A disk image will appear on your desktop; this contains the software that originally shipped with your Mac. To install an app, just drag it from the disk image to your hard drive.

## **38. iMOVIE FULL SCREEN AND OS X**

If you want to preview a short section of an iMovie running under OS X, don't bother switching to Full Screen mode. One of the quirks of the OS X version of iMovie is that a movie will always begin playing from the beginning rather than from the location of the playhead. To play from the position of the playhead, just stay in Normal view and preview your clip in the Viewer.

## **39. OS X MAIL IMPORT**

OS X's Mail application can't automatically import mail from Outlook Express unless that mail resides within the Identities folder that sits within the Microsoft User Data folder. In most cases, Outlook Express users will discover that their Identities folder is indeed at OS 9 Volume: Documents: Microsoft User Data. If you've moved that Identities folder and want to automatically import mail from Outlook Express, now might be a good time to put it back.

Alternatively, you can use the Browse button on the second screen of the Import Mailboxes utility to select the Identities folder containing your Outlook Express e-mail.

## **40. SOFTWARE UPDATE REDUX**

The OS X version of Software Update may not accurately reflect which updates are available when you first run it. For this reason, it's a good idea to click on the Update Now button a couple of times to see if additional updates appear.

## **41. MAILBOX LOCKED WARNING**

If you open OS X's Mail application only to be greeted by a message that indicates that such-and-such mailbox is locked, there's a reason. The Mail application will produce this error message if Mail was not properly shut

down (if you force-quit the application, for example), if you've changed the Mac's network connection with Mail open, or if two or more accounts are configured to place messages in the Personal Mailboxes Inbox.

You can usually work around this error by selecting the Open Anyway option and silently cursing Mail for not just opening the darned thing without this kind of intervention.

## 42. MAIL RULES

If you try to apply a rule to a large number of e-mail messages in OS X's Mail application, and the rule appears not to work with all the messages you've selected, try selecting fewer messages. Apple suggests that you choose no more than 256 messages when manually applying rules.

## 43. JUGGLING KEYBOARD COMMANDS

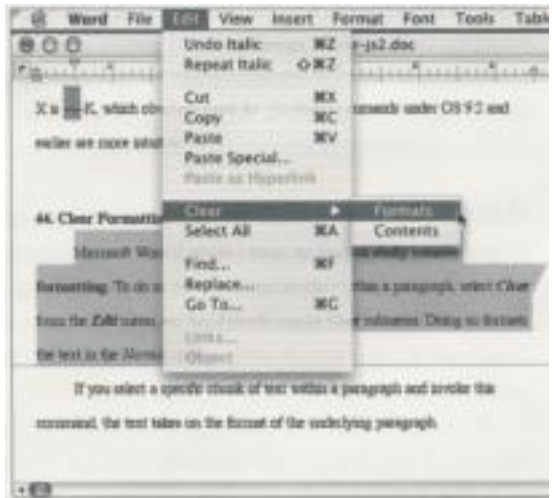
Outlook Express and Entourage 2001 users interested in upgrading to Microsoft Office X should prepare to abandon their habit of pressing  $\text{⌘-M}$  to send and receive all e-mail. Mac OS X reserves  $\text{⌘-M}$  for minimizing windows (to see this effect, press the yellow button in an OS X window). And what happened to the Make Alias command that's invoked when you press  $\text{⌘-M}$  in Mac OS 9.2 and earlier? Make Alias under OS X is  $\text{⌘-L}$ .

The new keyboard command for sending and receiving all mail in Entourage X is  $\text{⌘-K}$ , which obviously stands for "Keyboard commands under OS 9.2 and earlier are more intuitive."

## 44. CLEAR FORMATTING

Microsoft Word X includes a feature that lets you easily remove formatting. To do so, place your cursor anywhere within a paragraph, select Clear from the Edit menu, and then Formats from the Clear sub-menu. Doing so formats the text in the Normal style.

If you select a chunk of text within



It's easy to clear formatting in Microsoft Word X

a paragraph and invoke this command, the text takes on the format of the underlying paragraph.

## **45. APPLEWORKS 6.1 AND iDISK**

If you try to save an AppleWorks 6.1 document directly to your iDisk from within Mac OS X, that document may become corrupted. When you attempt to open it, you'll receive an error that reads "Unexpected End of File" and the file won't open.

To skirt this problem, first save the file to your Mac's hard drive and then copy it to your iDisk.

## **46. MOVING OMNIWEB ICONS**

OmniWeb, The Omni Group's Web browser for OS X, lets you customize the toolbar by either selecting Customize Toolbar from the Window menu or control-clicking on the tool bar and selecting Customize Toolbar from the contextual menu. However, once you've added an icon you like to the toolbar, how do you move it around?

Easy enough. Just hold down the ⌘ key while clicking on a toolbar icon to move it around.

# **H A R D W A R E / C O M P A T I B I L I T Y**

## **47. PC-FORMATTED ZIP DISKS AND OS X**

If you've mounted a PC-formatted Zip disk on your Mac running OS X, you can't save directly to that disk from an open Classic application. Instead, you must save the file to your hard disk and then copy the file to the Zip by drag and drop. Likewise, you can't use a Classic application's Open dialog box to open files on PC-formatted Zip disks. Again, you must first copy these files to your hard drive before opening them.

## **48. APPLE SYSTEM PROFILER MODEM CONFUSION**

Under Mac OS X, the Apple System Profiler may fail to list the Mac's modem. Where it should indicate a modem in the Network Overview section, Apple System Profiler might instead indicate an additional Ethernet device listed with an address of 00.00.00.00.00.00 and the entry "PointToPoint" in the Flags field.

If you see such an entry, don't worry: it doesn't mean your modem is broken. If you think your modem may be in dire straights, check it out with the Apple Hardware Test CD that came with your Mac.

## 49. MORE MODEM INFO

If OS X's Internet Connect application doesn't give you the kind of extensive information you need when your modem fails to connect properly, try this trick:

Open the Network system preference and choose Internal Modem from the Configuration pop-up menu. Click on the PPP tab and then click on the PPP Options button in the resulting window. Enable the Use Verbose Logging option, click on OK, and then click on Apply Now.

Open Internet Connect and choose Connection Log from the Windows menu. When you connect, a transcript of what's taking place during the modem connection appears in the Connection Log window. You can use this information to troubleshoot your modem connection.

## 50. DON'T PANIC

Should OS X crash with a kernel panic at start-up, Apple suggests that you try these remedies:

- Make sure your Mac has the latest firmware.
- Remove all peripheral devices except for your Mac's keyboard and mouse. If the Mac no longer panics once these devices are gone, add them back one at a time until the Mac malfunctions. Once you've isolated the funky device, check its manufacturer's Web site for OS X -compatible drivers.
- Remove any internal third-party upgrades such as RAM, accelerators, and PCI cards. Again, add them back in one at a time in an attempt to isolate the one causing the problem. If a RAM upgrade seems to be the culprit, contact its manufacturer to see if the RAM meets Apple's specifications.
- Boot from the OS X installation disc and run Disk Utility.

*Macworld* Contributing Editor CHRISTOPHER BREEN answers questions and provides tips monthly in *Macworld* and weekly in Macworld's weekly newsletter, available free at [www.macworld.com/newsletters](http://www.macworld.com/newsletters). These tips were culled from recent newsletter issues.



# Unix Tricks

This excerpt was adapted from *Learning the Unix Operating System*, fifth edition (O'Reilly, 2002), by JERRY PEEK, GRACE TODINO, and JOHN STRANG. A forthcoming edition, *Learning the Unix for Mac OS X* (O'Reilly, 2002), by DAVE TAYLOR and JERRY PEEK, will offer a concise introduction to Unix specifically designed for Mac OS X users.

Unix was invented more than 30 years ago for scientific and professional users who wanted a very powerful and flexible operating system. It's been significantly developed since then. Because Unix was designed for experts, it can be a bit overwhelming at first. But after you understand some Unix basics—outlined in this booklet—you'll start to appreciate some of the reasons to go beyond Mac OS X's Aqua interface and use the Unix bubbling beneath the surface.

## THE UNIX SHELL

Once you launch OS X's Terminal app, you're working with a program called a shell. The shell interprets command lines you enter, runs programs you ask for, and generally coordinates what happens between you and the Unix operating system. Common shells include Bourne (sh), Korn (ksh), and C (csh) shells, as well as bash and tcsh; the latter is Mac OS X's default shell.

For a beginner, differences between shells are slight. If you plan to work a lot with Unix, though, you should learn more about your shell and its special commands.

## THE SHELL PROMPT

When the system is ready to run a command, the shell outputs a prompt to tell you that you can enter a command line.

Shell prompts usually end with \$ or %. The prompt can be customized, though, so your own shell prompt may be different. By default, Mac OS X's shell ends with %.

## ENTERING A COMMAND LINE

Entering a command line at the shell prompt tells the computer what to do. Each command line includes the name of a Unix program. When you press the return key, the shell interprets your command line and executes the program.

The first word that you type at a shell prompt is always a Unix command (or program name). Like most things in Unix, program names are case sensitive; if the program name is lowercase (and most are), you must type it in lowercase. Some simple command lines have just one word, which is the program name.

**date** An example of a single-word command is `date`. Entering this command displays the current date and time:

```
% date
Wed Nov 7 06:25:44 PST 2001
%
```

As you type a command line, the system simply collects your keyboard input. Pressing the return key tells the shell that you've finished entering text and that it can run the program.

**who** Another simple command is `who`. It displays a list of each logged-on user's username, terminal number, and login time.

The `who` program can also tell you who is logged in at your terminal. The command line is `who am I`. This command line consists of the command (`who`), the program's name (`am I`) and arguments (`am I`):

```
% who am I
john tty1 Oct 6 08:26
```

The response shown in this example says that:

- "I am" John (actually, my username is john).
- I'm using terminal `tty1`.
- I logged in at 8:26 on the morning of October 6.

## RECALLING PREVIOUS COMMANDS

Modern Unix shells remember command lines you've typed before. They can even remember commands from previous login sessions. This handy feature can save you a lot of retyping common commands. As with many things in Unix, though, there are several different ways to do this; we don't have room to show and explain them all.

After you've typed and executed several command lines, try pressing the up-arrow key on your keyboard. You should see the previous command line after your shell prompt, just as you typed it before. Pressing the up-arrow key again recalls the previous command line, and so on. Also, as you'd expect, the down-arrow key will recall more-recent command lines.

To execute one of these remembered commands, just press the return key. (Your cursor doesn't have to be at the end of the command line.)

Once you've recalled a command line, you can also edit it. If you don't want to execute any remembered commands, cancel the command line by pressing control-C.

## CORRECTING A COMMAND LINE

What if you make a mistake in a command line? Suppose you typed `date` instead of `date` and pressed the return key before you realized your mistake. The shell would first ask if you meant to type `date-a`—a convenient fix! But that happens only if what you typed is close enough to what you meant to type that the shell can figure it out. If you type something more comprehensively wrong—say `ls-la` instead of `ls -la`—the shell will give you an error message:

```
% Is-Ia
Is-Ia: command not found
%
```

Don't be too concerned about getting error messages. Sometimes you'll get an error even if it appears that you typed the command correctly. This can be caused by typing control characters that are invisible on screen. Once the prompt returns, re-enter your command.

As we said earlier, most modern shells let you recall previous commands and edit command lines. If you'll do a lot of work at the shell prompt, it's worth learning these handy techniques. They take more time to learn than we can spend here, though—except to mention that, on those shells, the left-arrow and right-arrow keys may move your cursor along the command line to the point where you want to make a change. Here, let's concentrate on simple methods that work with all shells.

- If you see a mistake before you press return, you can use the erase key to erase and correct the mistake. The erase key differs from system to system and from account to account, and it can be customized. In Mac OS X, it's the left-arrow key. In many other Unix-based operating systems, it's control-H.

- The delete key may be used as the interrupt character instead of the erase character. This key is used to interrupt or cancel a command, and can be used in many (but not all) cases when you want to quit what you're doing. Another key combination that does the same thing is control-C.

Other common control characters are:

**control-U:** Erases the whole input line; you can start over.

**control-S:** Pauses output from a program that's writing to the screen. This can be confusing; we don't recommend using control-S but want you to be aware of it.

**control-Q:** Restarts output after a pause by control-S.

**control-D:** Used to signal end-of-input for some programs and return you to a shell prompt. If you type control-D at a shell prompt, it may close your terminal window or log you out of the Unix system.

## LOGGING OUT

To end a Unix session, you must log out. You can log out by entering the command `exit` at a shell prompt. (In many cases, the command `logout` will also work.) Depending on your shell, you may also be able to log out simply by typing control-D.

## SYNTAX OF UNIX COMMAND LINES

Unix command lines can be simple, one-word entries, such as the `date` command. They can also be more complex; you may need to type more than the command or program name.

A Unix command may or may not have *arguments*. An argument can be an option or a filename. The general format for Unix command lines is:

**command option(s) filename(s)**

There isn't a single set of rules for writing Unix commands and arguments, but you can use these general rules in most cases:

- Enter commands in lowercase.
- *Options* modify the way in which a command works. Options are often single letters prefixed with a dash (-, also called a "hyphen" or "minus sign") and set off by any number of spaces or tabs. Multiple options in one command line can be set off individually (such as `-a -b`). In some cases, you can combine them after a single dash (such as `-ab`), but most commands' documentation doesn't tell you whether this will work; you'll have to try it.

Some commands, including those on Linux systems, also have options made from complete words or phrases and starting with two dashes, like `--delete` or `--confirm-delete`. When you enter a command line, you can use this option style, the single-letter options (which all start with a single dash), or both.

- A *filename* is the name of a file that you want to use. Most Unix programs also accept multiple filenames, separated by spaces. If you don't enter a filename correctly, you may get a response such as "*filename*: no such file or directory" or "*filename*: cannot open."

Some commands, such as `telnet` and `who`, have arguments that aren't filenames.

- You must type spaces between commands, options, and filenames.
- Options come before filenames.
- In a few cases, an option has another argument associated with it; type this special argument just after its option. Most options don't work this way, but you should know about them. The `sort` command is an example of this: you can tell `sort` to write the sorted text to a filename given after its `-O` option. In the following example, `sort` reads the file

sortme (given as an argument), and writes to the file sorted (given after the -O option):

```
% sort -O sorted -n sortme
```

We also used the -n option in that example. But -n is a more standard option; it has nothing to do with the final argument sort me on that command line. So, we also could have written the command line this way:

```
% sort -n -O sorted sort me
```

- Command lines can have other special characters. They can also have several separate commands. For instance, you can write two or more commands on the same command line, each separated by a semicolon (;). Commands entered this way are executed one after another by the shell.

## A SAMPLE COMMAND

Unix has a lot of commands! Don't try to memorize all of them. In fact, you'll probably need to know just a few commands and their options. As time goes on, you'll learn these commands and the best way to use them for your job.

Let's look at a sample Unix command. The ls program displays a list of files. You can use it with or without options and arguments. If you enter:

```
%ls
```

you'll see a list of filenames. But if you enter:

```
% ls-l
```

there'll be an entire line of information for each file. The -l option changes the normal ls output to a long format. You can also get information about a particular file by using its name as the second argument. For example, to find out about a file called *chap1*, enter:

```
% ls -l chap1
```

Many Unix commands have more than one option. For instance, ls also has the -a (all) option for listing hidden files. You can use multiple options in either of these ways:

```
% ls -a -l
```

```
% ls -al
```

As we said earlier, you must type one space between the command name and the dash that introduces the options. If you enter ls-ai, the shell will say "ls-al: command not found."

## ENTERING A FEW COMMANDS

The best way to get used to Unix is to enter some commands. To run a command, type the command and then press the return key. Remember that almost all Unix commands are typed in lowercase.

- Get today's date: Enter **date**
- List logged-in users: Enter **who**
- Obtain more information about users: Enter **who -u** or **finger** or **w** . Find out who is at your terminal: Enter **who am I**
- Enter two commands in the same line: Enter **who am i;date**
- Mistype a command: Enter **woh**

In this session, you've tried several simple commands and seen the results on the screen.

## THE UNIX FILE SYSTEM

A *file* is the unit of storage in Unix, as in most other systems. A file can hold anything: text (a report you're writing, a to-do list), a program, digitally encoded pictures or sound, and so on. All of those are just sequences of raw data until they're interpreted by the right program.

In Unix, files are organized into directories. A *directory* is actually a special kind of file where the system stores information about other files. You can think of a directory as a place, so that files are said to be contained *in* directories and you are said to work *inside* a directory. (In Finder terms, a Unix directory is the equivalent of a folder.)

This section introduces the Unix file system. Later sections show how you can look in files and protect them.

## YOUR HOME DIRECTORY

When you log in to Unix, you're placed in a directory called your *home directory*. This directory contains the files you use almost every time you log in. As you'll see, you can also store your own directories within your home directory. Like folders in a file cabinet, this is a good way to organize your files.

## YOUR WORKING DIRECTORY

Your *working directory* (also called your current directory) is the directory you're currently working in. Every time you log in, your home directory is your working directory. You may change to another directory, in which case the directory you move to becomes your working directory.

Unless you tell Unix otherwise, all commands that you enter apply to the files in your working directory. In the same way, when you create files, they're created in your working directory unless you specify another directory. For instance, if you type the command `pico report`, the Pico editor is started on a file named *report* in your working directory. But if you type a command such as `pico /home/joan/report`, a report file is edited

in a different directory without changing your working directory. You'll learn more about this when we cover path names later in this section.

If you have more than one terminal window open, or if you're logged in on several terminals at the same time, each session has its own working directory. Changing the working directory in one session doesn't affect others.

## THE DIRECTORY TREE

All directories on a Unix system are organized into a hierarchical structure that you can imagine as a family tree. The parent directory of the tree (the directory that contains all other directories) is known as the *root directory* and is written as a forward slash (*f*).

The root contains several directories. The figure on page 25 shows a visual representation of the top of a Unix filesystem tree: the root directory and some directories under the root. *bin*, *etc*, *users*, *tmp*, and *usr* are some of the *subdirectories* (child directories) of the root directory. These subdirectories are fairly standard directories; they usually contain specific kinds of system files. For instance, *bin* contains many Unix programs. In addition to all of these Unix-specific directories (most of which are hidden when you're in Mac OS X's Finder), the root directory of your hard drive will contain the various files and folders you can see in the Finder, including your Applications folder.

To specify a file or directory location, write its *pathname*. A pathname is like the address of the directory or file in the Unix filesystem. We'll look at pathnames in a moment.

On a basic Unix system, all files in the filesystem are stored on disks connected to your computer. It isn't always easy to use the files on someone else's computer or for someone on another computer to use your files. Mac OS X, however, also allows you to access files over the *networked filesystem*. Networked file systems make a remote computer's files appear as if they were part of your computer's directory tree. For instance, a computer in Los Angeles might have a directory named *boston* with some of the directory tree from a company's computer in Boston. Or individual users' home directories may come from various computers, but all be available on your computer as if they were local files. Once you've mounted a volume via the Connect To Server command in the Finder's Go menu, you can find those volumes in Terminal within the */Volumes* directory.

**Absolute Pathnames** The Unix filesystem organizes its files and directories in an inverted tree structure with the root directory at the top. An *absolute pathname* tells you the path of directories you must travel to

get from the root to the directory or file you want. In a path name, put slashes between the directory names.

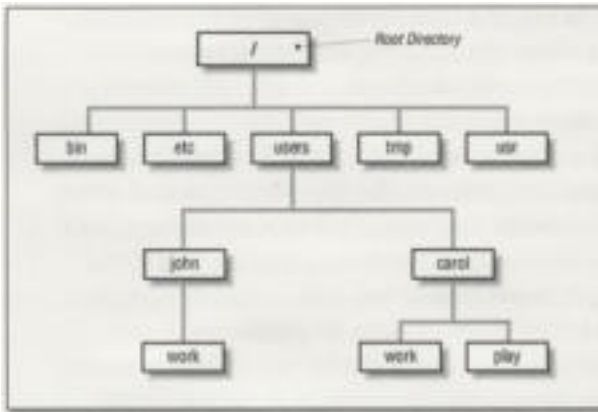
For example, `/users/john` is an absolute pathname. It locates one (*only one*) directory. Here's how:

- The root is the first `/`
- The directory `users` (a subdirectory of root)
- The directory `john` (a subdirectory of `users`)

Be sure that you do not type spaces anywhere in the pathname.

If `john` has a subdirectory inside it named `work`, its absolute pathname would be `/users/john/work`.

The root is always indicated by the slash at the start of the pathname. In other words, *an absolute pathname always starts with a slash*.



**Example of a directory tree.**

## Relative Pathnames

You can also locate a file or directory with a *relative pathname*. A relative pathname gives the location relative to your working directory. Unless you use an absolute pathname (starting with a slash), Unix assumes that

you're using a relative

pathname. Like absolute pathnames, relative path names can go through more than one directory level by naming the directories along the path.

For example, if you're currently in the `users` directory, the relative path name to Carol's user directory is simply `carol`. The relative pathname to the `play` directory inside the `carol` directory is `carol/play`.

Notice that neither pathname in the previous paragraph starts with a slash. That's what makes them relative pathnames! Relative pathnames start at the working directory, not the root directory. In other words, *a relative pathname never starts with a slash*.

**Pathname Puzzle** Here's a short but important question. The previous example explains the relative pathname `carol/play`. What do you think Unix would say about the path name `/carol/play`?

Unix would say "No such file or directory." Why? (Please think about that before you read more. It's very important and it's one of the most

common beginner's mistakes.) Here's the answer: Because it starts with a slash, the path name `/carol/play` is an absolute path name that starts from the root. It says to look in the root directory for a subdirectory named *carol*. But there is no subdirectory named *carol* one level directly below the root, so the path name is wrong. The only absolute path name to the play directory is `/users!carol/play`.

**Relative Pathnames Up** You can go up the tree with the shorthand `..` (two periods) for the parent directory. As you saw earlier, you can also go down the tree by using subdirectory names. In either case (up or down), separate each level by a slash.

Say your working directory is the work subdirectory of *carol*. Then there are two path names for play, another subdirectory of carol. You already know how to write the absolute path name: `/users!carol/play`. You can also go up one level (with `..`) to *carol*, then go down the tree to *play*.

The relative path name would be `play`. It would be wrong to give the relative address as `carol/play`. Using `carol/play` would say that *carol* is a subdirectory of your working directory instead of what it is in this case, the parent directory.

Absolute and relative path names are totally interchangeable. Unix programs simply follow whatever path you specify to wherever it leads. If you use an absolute path name, the path starts from the root. If you use a relative path name, the path starts from your working directory. Choose whichever is easier at the moment.

## CHANGING YOUR WORKING DIRECTORY

Once you know the absolute or relative pathname of a directory where you'd like to work, you can move up and down the Unix directory tree to reach it.

**pwd** To find which directory you're currently in, use `pwd` (print working directory). The `pwd` command takes no arguments and prints the absolute path name of your working directory.

```
%pwd
/users/john
$
```

**cd** You can change your working directory to any directory (including another user's directory, if you have permission) with the `cd` (change directory) command.

The `cd` command has the form:

```
cd pathname
```

The argument is an absolute or a relative pathname (whichever is easier) for the directory you want to change to:

```
% cd /users/carol
% pwd
/users/carol
% cd work
% pwd
/users/carol/work
%
```

Here's a time-saver: the command `cd`, with no arguments, takes you to your home directory from wherever you are in the file system.

Note that you can change only to another directory. You cannot use `cd` to get to a filename. If you try, your shell gives you an error message.

## FILES IN THE DIRECTORY TREE

A directory can hold subdirectories. And of course, a directory can hold files. Path names to files are made the same way as pathnames to directories. As with directories, files' path names can be absolute (starting from the root directory) or relative (starting from the working directory). For example, if your working directory is called *users*, the relative path name to the *work* directory below would be `john/work`. The relative pathname to a file named *ch1* would be `john/ch1`.

## LISTING FILES WITH `ls`

To use the `cd` command, you must decide which entries in a directory are subdirectories and which are files. The `ls` command lists entries in the directory tree so you can see which is which.

When you enter the `ls` command, you'll get a listing of the files and subdirectories contained in your working directory. The syntax is:

```
ls option(s) directory-and-filename(s)
```

If you've just logged in for the first time, entering `ls` without any arguments may seem to do nothing. This isn't surprising—you haven't made any files in your working directory. If you have no files, nothing is displayed; you'll simply get a new shell prompt:

```
% ls
%
```

But if you've already made some files or directories in your account, those names are displayed. The output depends on what's in your directory. The screen should look something like this:

```
% ls
ch1 ch10 ch2 ch3 intro
%
```

`ls` has a lot of options that change the information and display format.

The `-a` option (for *all*) is guaranteed to show you some more files, as in the following example:

```
% ls -a
.exrc ch1 ch2 intra .. .profile ch10 ch3
%
```

When you use `ls -a`, you'll always see at least two entries with the names `.` (period) and `..` (two periods). As mentioned earlier, `..` is always the relative path name to the parent directory. A single dot always stands for its working directory; this is useful with commands like `cpo`. There may also be other files, such as `.profile` or `.exrc`. Any entry whose name begins with a dot is hidden—it's listed only if you use `ls -a`.

To get more information about each item that `ls` lists, add the `-l` option. (That's a dash and a lowercase *L* for "long.") This option can be used alone, or in combination with `-a`, as seen in the figure on page 29.

The long format provides the following information about each item:

**Total** *n* amount of storage used by everything in this directory. (This is measured in *blocks*. On many systems, a full block holds 1,024 bytes. A block can also be partly full.)

**Type** Tells whether the item is a directory (`d`) or a plain file (`-`). (There are other less common types that we don't explain here.) **Access Modes** Specifies three types of users (yourself, your group, all others) who are allowed to read (`r`), write (`w`), or execute (`x`) your files. We'll say more about this in a moment.

**Links** The number of files or directories linked to this one. (This isn't the same sort of link as in a Web page. We don't discuss file system links in this little book.)

**Owner** The user who created or owns this file or directory.

**Group** The group that owns the file or directory.

**Size (in bytes)** The size of the file or directory. (A directory is actually a special type of file. Here, the "size" of a directory is of the directory file itself, not of all the files in that directory.)

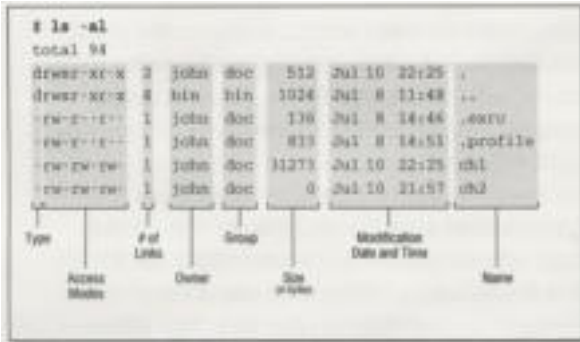
**Modification Date and Time** When the file was last modified or the directory contents were last changed (when something in the directory was added, renamed, or removed). If an entry was modified more than six months ago, `ls` shows the year instead of the time.

**Name** The name of the file or directory.

Notice especially the columns that list the owner and group of the files, and the access modes (also called *permissions*). The person who creates a file is its owner; if you've created any files (or the systems staff did it for you), this column should show your username.

You also belong to a group, set by the person who created your account. Files you create are marked either with the name of your group or, in some cases, the group that owns the directory.

The permissions show who can read, write, or execute the file or directory; we explain what that means in a moment. The permissions have ten characters. The first character shows the file type (d for directory



or - for a plain file). The other characters come in groups of three. The first group, characters 2-4, shows the permissions for the file's owner, which is you if you created the file. The second

### Output from ls -al

group, characters 5-7, shows permissions for other members of the file's group. The third group, characters 8-10, shows permissions for all other users.

For example, the permissions for *.profile* are *-rw-r--r--*, so it's a plain file. The owner, *john*, has both read and write permissions. Other users who belong to the file's group, *doc*, as well as all other users of the system, can only read the file; they don't have write permission, so they can't change what's in the file. No one has execute (x) permission, which should only be used for executable files (files that hold programs).

In the case of directories, x means the permission to access the directory, for example, to run a command that reads a file there or to use a subdirectory. Notice that the two directories shown in the example are executable (accessible) by *john*, by users in the *doc* group, and by everyone else on the system. A directory with write (w) permission allows deleting, renaming, or adding files within the directory. Read (r) permission allows listing the directory with *ls*.

You can use the *chmod* command to change the permissions of your files and directories.

If you need to know only which files are directories and which are executable files, you can use the *-F* option.

If you give the pathname to a directory, *ls* lists the directory but it does *not* change your working directory. The *pwd* command in the

following example shows this:

```
% ls -F /users/andy
calendar goals ideas/
ch2 guide/ testpgm*
%pwd
/etc
%
```

ls -F puts a slash at the end of each directory name. (The directory name doesn't really have a slash in it; that's just the shortcut ls -F uses to identify a directory.) In our example, *guide* and *ideas* are directories. You can verify this by using ls -l and noting the *d* in the first field of the output. Files with an execute status (x), such as programs, are marked with an asterisk (\*). The file *testpgm* is an executable file. Files that aren't marked are not executable.

ls -R ("recursive") lists a directory and all its subdirectories. This can make a very long list, especially when you list a directory near the root! (Piping the output of ls to a pager program solves this problem, as we'll show later on.) You can combine other options with -R : for instance, ls -RF marks each directory and file type.

## EXPLORING THE FILE SYSTEM

You're now equipped to explore the file system with cd, ls, and pwd: Take a tour of the directory system, hopping one level or many at a time, with a mixture of cd and pwd commands.

- Go to your home directory: Enter cd
- Find your working directory: Enter pwd
- Change to new working directory with its absolute path name: Enter cd /etc
- List files in new working directory: Enter ls
- Change directory to root and list it in one step. (Use the command separator, a semicolon.): Enter cd /; ls
- Find your working directory: Enter pwd
- Change to a subdirectory; use its relative path name: Enter cd usr
- Find your working directory: Enter pwd
- Change to a subdirectory: Enter cd bin
- Find your working directory: Enter pwd
- Give a wrong path name: Enter cd xqk
- List files in another directory: Enter ls /bin
- Find your working directory (notice that ls didn't change it): Enter pwd .
- Return to your home directory: Enter cd

# INPUT AND OUTPUT

What happens if you don't give a filename argument on a command line? Most programs will take their input from your keyboard instead (after you press the first return to start the program running, that is). Your keyboard is the program's *standard input*.

As a program runs, the results are usually displayed on your screen. The screen is the program's *standard output*.

So, by default, each of these programs takes its input from the standard input and sends the results to the standard output.

These two default cases of input/output (I/O) can be varied. This is called *I/O redirection*.

If a program doesn't normally read from files, but reads from its standard input, you can give a filename by using the less-than symbol (<) operator. For example, the Unix mail program normally reads the message to send from your keyboard. Here's how to use the input redirection operator to mail the contents of the file *to\_do* to *bigboss@corp.xyz*:

```
% mail bigboss@corp.xyz < to_do
%
```

If a program writes to its standard output, which is normally the screen, you can make it write to a file instead by using the greater-than symbol (>) operator. The pipe operator (|) sends the standard output of one program to the standard input of another program. Input/output redirection is one of the most powerful and flexible Unix features. We'll take a closer look at it.

## PUTTING TEXT IN A FILE

Instead of always letting a program's output come to the screen, you can redirect output into a file. This is useful when you'd like to save program output or when you put files together to make a bigger file.

**cat** Short for "concatenate," **cat** reads files and outputs their contents one after another, without stopping.

To display files on the standard output (your screen), use:

```
cat file(s)
```

For example, let's display the contents of the file */etc/passwd*. This system file describes users' accounts. (Your system may have a more complete list somewhere else.)

```
% cat /etc/passwd
nobody:*:-2:-2:Unprivileged User:/nohome:/noshell
root:*:0:0:System Administrator:/var/root:/bin/tcsh
%
```

You cannot go back to view the previous screens, as you can when you use a pager program such as `less` (unless you're using a terminal window with a scrollbar, that is). `cat` is used mainly with redirection, as we'll see in a moment.

By the way, if you enter `cat` without a filename, it tries to read from the keyboard. You can get out by pressing control-D once.

**The > operator** When you add `> filename` to the end of a command line, the program's output is diverted from the standard output to the named file. The `>` symbol is called the *output redirection operator*.

When you use the `>` operator, be careful not to accidentally overwrite a file's contents. Your system may let you redirect output to an existing file. If so, the old file will be deleted (or, in Unix lingo, "clobbered"). Be careful not to overwrite a needed file!

Let's use `cat` with this operator. The file contents that you'd normally see on the screen (from the standard output) are diverted into another file, which we'll then read using `cat` (without any redirection):

```
% cat /etc/passwd > password
% cat password
nobody:*:-2:- 2:Unprivileged User:/nohome:/noshell
root:*:0:0:System Administrator:/var/root:/bin/tcsh
%
```

An earlier example showed how `cat /etc/passwd` displays the file `/etc/passwd` on the screen. The example here adds the `>` operator; so the output of `cat` goes to a file called `password` in the working directory. Displaying the file `password` shows that its contents are the same as the file `/etc/passwd` (the effect is the same as the copy command `cp /etc/passwd password`).

You can use the `>` redirection operator with any program that sends text to its standard output—not just with `cat`. For example:

```
% who > users
% date > today
% ls
password today users ...
```

We've sent the output of `who` to a file called `users` and the output of `date` to the file named `today`. Listing the directory shows the two new files. Let's look at the output from the `who` and `date` programs by reading these two files with `cat`:

```
% cat users
john console Jan 28 13:10
john tty1 Jan 29 12:38
% cat today
```

```
Wed Nov 7 06:25:09 PST 2001
```

```
%
```

You can also use the `cat` program and the `>` operator to make a small text file. We told you earlier to press control-D if you accidentally enter `cat` without a filename. This is because the `cat` program alone takes whatever you type on the keyboard as input. Therefore, the command:

```
cat> filename
```

takes input from the keyboard and redirects it to a file. Try the following example:

```
% cat> to do
```

```
Finish report by noon
```

```
Lunch with Xannie
```

```
Swim at 5:30
```

```
AD
```

```
%
```

`cat` takes the text that you typed as input (in this example, the three lines that begin with `Finish`, `Lunch`, and `Swim`), and the `>` operator redirects it to a file called `to_do`. Press control-D *once*, on a new line by itself, to signal the end of the text. You should get a shell prompt.

You can also create a bigger file from smaller files with the `cat` command and the `>` operator. The form:

```
cat file1 file2 > newfile
```

creates a file *newfile*, consisting of *file1* followed by *file2*.

```
% cat today to_do> diary
```

```
% cat diary
```

```
Wed Nov 7 06:25:09 PST 2001
```

```
Finish report by noon
```

```
Lunch with Xannie
```

```
Swim at 5:30
```

```
%
```

**The» operator** You can add more text to the end of an existing file, instead of replacing its contents, by using the» (append redirection) operator. Use it as you would the `>` (output redirection) operator. So:

```
cat file2 » file1
```

appends the contents of *file2* to the end of *file1*. For an example, let's append the contents of the file *users*, and also the current date and time, to the file *diary*. Then we display the file:

```
% cat users» diary
```

```
% date» diary
```

```
% cat diary
```

```
Wed Nov 7 06:25:09 PST 2001
```

```
finish report by noon
Lunch with Xannie
Swim at 5:30
john console Jan 28 13:10
john tty1 Jan 29 12:38
Wed Nov 7 06:25:09 PST 2001
%
```

Unix doesn't have a redirection operator that adds text to the beginning of a file. You can do this by storing the new text in a temporary file and then using a text editor to read the temporary file into the start of the file you want to edit. You also can do the job with a temporary file and redirection. Maybe you'd like each day's entry to go at the beginning of your *diary* file. Simply rename *diary* to something like *temp*. Make a new *diary* file with today's entries, then append *temp* (with its old contents) to the new *diary*. For example:

```
% mv diary temp
% date> diary
% cat users» diary
% cat temp» diary
% rm temp
```

## PIPES AND FILTERS

We've seen how to redirect input from a file and output to a file. You can also connect two programs together so that the output from one program becomes the input of the next program. Two or more programs connected in this way form a *pipe*. To make a pipe, put a vertical bar (|) on the command line between two commands. When a pipe is set up between two commands, the standard output of the command to the left of the pipe symbol becomes the standard input of the command to the right of the pipe symbol. Any two commands can form a pipe, as long as the first program writes to standard output and the second program reads from standard input.

When a program takes its input from another program, performs some operation on that input, and writes the result to the standard output (which may be piped to yet another program), it is referred to as a *filter*. A common use of filters is to modify output. Just as a common filter culls unwanted items, Unix filters can restructure output.

Most Unix programs can be used to form pipes. Some programs that are commonly used as filters are described in the next sections. Note that these programs aren't used only as filters or parts of pipes. They're also useful on their own.

**grep** The grep program searches a file or files for lines that have a certain pattern. The syntax is:

```
grep "pattern" file(s)
```

grep's name derives from the ed (a Unix line editor) command g/re/p, which means "globally search for a regular expression and print all lines containing it." A *regular expression* is either some plain text (a word, for example) and/or special characters used for pattern matching. When you learn more about regular expressions, you can use them to specify complex patterns of text.

The simplest use of grep is to look for a pattern consisting of a single word. It can be used in a pipe so that only those lines of the input files containing a given string are sent to the standard output. But let's start with an example reading from files: searching all files in the working directory for a word—say, *Unix*. We'll use the wildcard \* to quickly give grep all filenames in the directory.

```
% grep "Unix" *  
ch01:Unix is a flexible and powerful operating system  
ch01:When the Unix designers started work, little did  
ch05:What can we do with Unix?  
%
```

When grep searches multiple files, it shows the filename where it finds each matching line of text. Alternatively, if you don't give grep a filename to read, it reads its standard input; that's the way all filter programs work:

```
% ls -l | grep "Aug"  
-rw-rw-rw- 1 john doc 11008 Aug 6 14:10 ch02  
-rw-rw-rw- 1 john doc 8515 Aug 6 15:30 ch07  
-rw-rw-r- 1 john doc 2488 Aug 15 10:51 intra  
-rw-rw-r- 1 carol doc 1605 Aug 23 07:35 macros  
%
```

First, the example runs `ls -l` to list your directory. The standard output of `ls -l` is piped to `grep`, which outputs only lines that contain the string *Aug* (that is, files that were last modified in August). Because the standard output of `grep` isn't redirected, those lines go to the terminal screen.

`grep` options let you modify the search.

## SOME GREP OPTIONS

- v Print all lines that do not match pattern.
- n Print the matched line and its line number.
- l (lowercase L) Print only the names of files with matching lines.
- c Print only the count of matching lines.

-l Match either upper- or lowercase.

Next, let's use a regular expression that tells grep to find lines with carol, followed by any number of characters, even none (abbreviated in a regular expression as .\*), then followed by Aug. (Note that the regular expression for "zero or more characters"- .\* -is different from the corresponding filename \* wildcard. We can't cover regular expressions in enough depth here to explain the difference; for more information, see *Mastering Regular Expressions* (O'Reilly, 1997) or *Transform HTML with Regular Expressions*

<<http://www.macworld.com/1998/11/create/4549.html>>.

```
% Is -lI grep "carol.*Aug"
-rw-rw-r- 1 carol doc 1605 Aug 23 07:35 macros
%
```

**sort** The sort program arranges lines of text alphabetically or numerically. The following example sorts the lines in the file *food* alphabetically. sort doesn't modify the file itself; it reads the file and writes the sorted text to the standard output.

```
% sort food
Afghani Cuisine
Bangkok Wok
Big Apple Deli
Isle of Java
Mandalay
Sushi and Sashimi
Sweet Tooth
Tio Pepe's Peppers
```

By default, sort arranges lines of text alphabetically. Many options control the sorting; and the following table lists some of them.

## SOME SORT OPTIONS

- n Sort numerically (example: 10 sorts after 2), ignore blanks and tabs.
- r Reverse the sorting order.
- f Sort upper- and lowercase together.
- +x Ignore first x fields when sorting.

More than two commands may be linked up into a pipe. Taking a previous pipe example using grep, we can further sort the files modified in August by order of size. The following pipe uses the commands ls, grep, and sort:

```
% ls -l | grep "aug" | sort +4n
-rw-rw-r- 1 carol doc 1605 Aug 23 07:35 macros
-rw-rw-r- 1 john doc 2488 Aug 15 10:51 intro
-rw-rw-rw- 1 john doc 8515 Aug 6 15:30 ch07
-rw-rw-rw- 1 john doc 11008 Aug 6 14:10 ch02
%
```

This pipe sorts all files in your directory modified in August by order of size, and prints them to the terminal screen. The sort option +4n skips four fields (fields are separated by blanks), then sorts the lines in numeric order. So, the output of ls, filtered by grep, is sorted by the file size (this is the fifth column, starting with 1605). Both grep and sort are used here as filters to modify the output of the <ls -l command. If you wanted to email this listing to someone, you could add a final pipe to the mail program. Or you could print the listing by piping the sort output to your printer command (either lp or lpr).

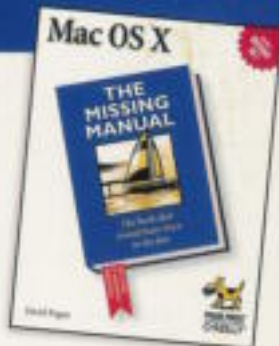
Copyright © 2002 O'Reilly & Associates, Inc

*Mac OS X: The Missing Manual* is  
the #1 bestselling book on its topic.  
Here's what the reviewers say...



"A must-have resource for all migrating  
or new Mac OS X users."

—*Netsurfer Digest* Jan 11, 2002



"There is a series of 'Missing Manuals' which  
are so good, I believe Apple should include them  
with every new Macintosh. This is no hype."

—*David L. Davis, macnj.org, Feb 2002*

"All in all, this is one of the most painless and easiest ways  
to learn the Macintosh there is. I highly recommend it."

—*Timothy Arends, Bytelines, Feb 2002*

"David Pogue's *Mac OS X: The Missing Manual* is honest,  
thorough and to the point. It really does cover aspects of using  
OS X that are essential but may not be immediately obvious—  
even to experienced Mac users...A very good choice."

—*Mark Sealey, www.thinksecret.com, Jan 14, 2002*



**20% off**  
discount on  
order form