SCO® Open Desktop®/ SCO Open Server™ User's Guide

Date: 16 April 1993
Document Version: 3.0.0A

What to read

Like SCO® Open Desktop®/SCO Open Server™, this book is designed to meet the needs of many different people. **You only need to read those parts of this book that apply to you:**

- If you have used icons, menus, and windows before, just read Chapters 1 and 4.
- For basics on using a mouse, icons, menus, windows, files, and directories, read Chapters 2 and 3.
- For information about electronic mail or customizing your Desktop, read Chapter 5 or 6.

Working in the graphical environment is covered in the first six chapters. The remainder of the book is devoted to advanced topics:

- working from a command line (Chapter 7)
- working with UNIX® commands and files (Chapters 8 through 13)
- working with DOS commands and files (Chapters 14 through 18)
- working with network commands and resources (Chapters 19 through 24)

SCO Open Desktop/SCO Open Server provides a simple, intuitive way for you to work with your computer. The Desktop is a graphical workspace that you can adapt to your own way of working. Files and programs are represented by small pictures, or *icons*. To look at a file or run a program, you just point to its icon with a mouse and click twice. Drag a file icon to move it to a new location, or drop it on a Print icon to print the file. Files and programs are displayed in *windows*, which you can move around and stack up like papers on your desk. *Menus* in each window list what you can do in that window; no need to remember commands, just select actions from a menu.

Beneath the Desktop, hidden from view, is a complex package of sophisticated software that manages the computer and connects it with other computers. Although you may rarely need to do so, the UNIX and DOS windows give you the ability to issue commands directly to this underlying operating system and networking software.

Assumptions

This guide assumes you are using the software, but that someone else is taking care of, or "administering," it. That is the normal arrangement when SCO Open Desktop/SCO Open Server (or any UNIX-based system) is set up for several users. If your software is set up for a single user, then you are also the *system administrator* and you should familiarize yourself with the material in the appropriate administrator's guides (listed in the following section).

This guide assumes that SCO Open Desktop/SCO Open Server is installed and running on your system. If it is not, see your system administrator or the *SCO Open Desktop/SCO Open Server Installation and Upgrade Guide*.

This guide does *not* assume you know anything about the UNIX system, or even about working with computers. See your system administrator for guidance on site-specific basics (such as how to use your particular hardware).

Where to find more information

This guide presents the basic information you need to begin working with this product. More information about using this product is available on the screen. Chapter 4 gives instructions for accessing online documentation.

For a guided introduction to the software, read the *SCO Open Desktop/SCO Open Server Tutorial*.

For installation, configuration, and troubleshooting information, see these SCO Open Systems Software books:

Installation and Upgrade Guide
Hardware Configuration Guide
Graphical Environment Administrator's Guide
System Administrator's Guide
(including *Performance and Troubleshooting*)
Release Notes
Open Server System Administrator's Guide

See also these *online* books:

User and Administrator Manual Pages
Features and Limitations
Standards Conformance

The *Release Notes* and *Features and Limitations* contain new information and corrections added after the other documentation was published. *User and Administrator Manual Pages* contains detailed information about each UNIX command; see "More information about commands" in Chapter 7 for instructions on viewing manual pages.

Instructions for applications you run on this software are provided by the suppliers of those applications. Additional information about DOS commands can be found in your DOS manuals.

Bookshelves full of information about the UNIX operating system and the X Windows System have been published. For a fuller discussion of any UNIX or X Windows topic, contact your technical bookstore or library (or SCO Support) for specific bibliographical references.

Notational conventions in this book

This book uses the following conventions:

- Onscreen **buttons**, menu names, and menu items appear in boldface type.
- *Directories* and *filenames* are shown in italic type.
- **Commands** that should be entered exactly as shown appear in boldface type.
- ***Variables*** to commands appear in italic type.
- Items contained within square brackets " [ ] " are optional.
- Ellipses " . . . " indicate that more than one element may be placed on the command line.

- Keys to be pressed are shown in the ⟨ and ⟩ brackets. For example, the delete key is shown as ⟨Del⟩.
- Sample commands do not explicitly state that you must press ⟨Enter⟩ to activate the command. Pressing ⟨Enter⟩ at the end of a command line entry is assumed.

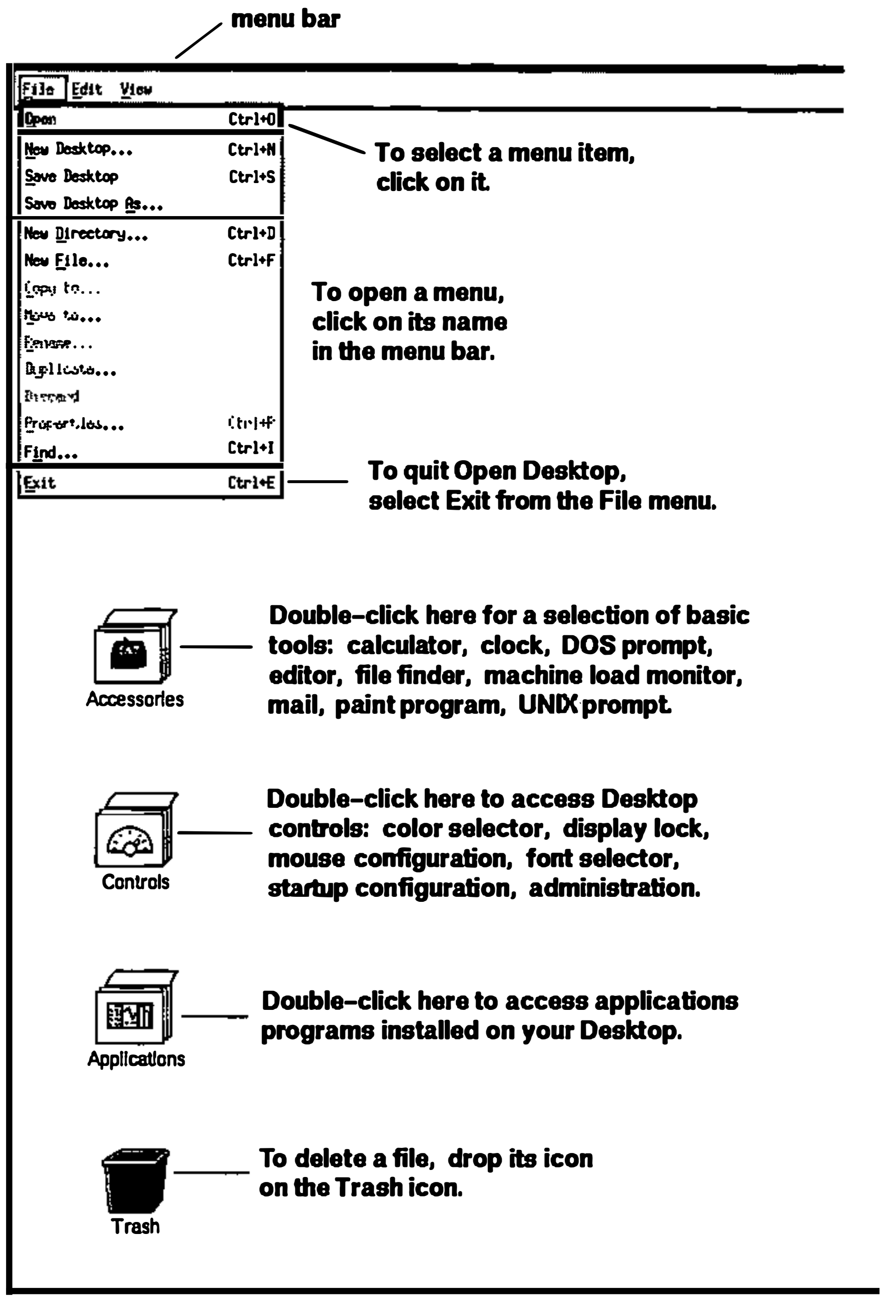
Chapter 1

The Desktop at a glance

Everything you need to know to start using SCO Open Desktop/SCO Open Server is in this short chapter. If you have used a computer with a graphical interface before, the following quick-reference illustrations are probably all you need to get going.

NOTE For more on the basics, read Chapters 2 through 6 or use the online help (Chapter 4 tells you how). If you have never used a mouse, please read "Using the mouse" (page 12) now.

Once you understand how to use the basic components of the Desktop, you can use new tools and applications almost immediately. Controls, icons, windows, and menus all work the same way everywhere on the Desktop.



To select a menu item, click on it.

To open a menu, click on its name in the menu bar.

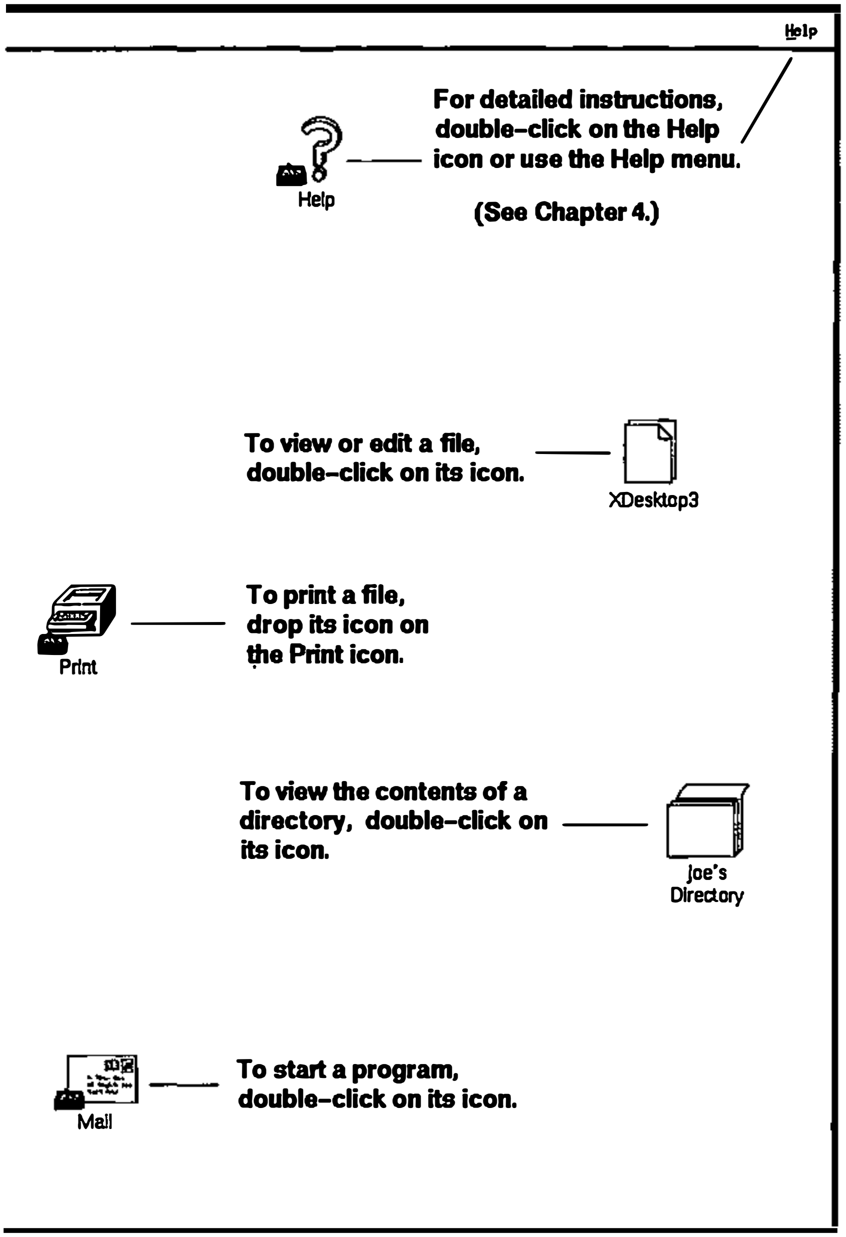
To quit Open Desktop, select Exit from the File menu.

Double-click here for a selection of basic tools: calculator, clock, DOS prompt, editor, file finder, machine load monitor, mail, paint program, UNIX prompt.

Double-click here to access Desktop controls: color selector, display lock, mouse configuration, font selector, startup configuration, administration.

Double-click here to access applications programs installed on your Desktop.

To delete a file, drop its icon on the Trash icon.



For detailed instructions, double-click on the Help icon or use the Help menu.

(See Chapter 4.)

To view or edit a file, double-click on its icon.

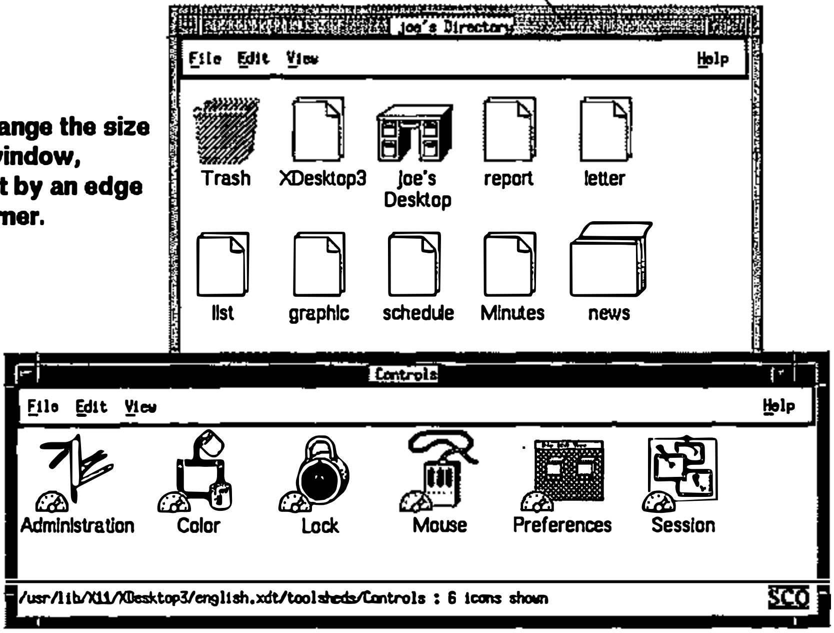
To print a file, drop its icon on the Print icon.

To view the contents of a directory, double-click on its icon.

To start a program, double-click on its icon.

To move a window, drag it by its title bar.

To change the size of a window, drag it by an edge or corner.



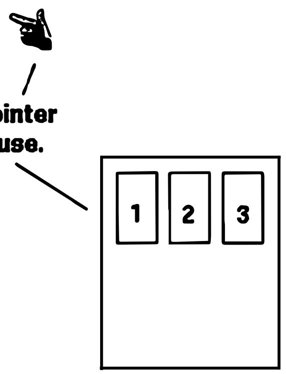
/usr/lib/X11/XDesktop3/english.xdt/toolsheds/Controls : 6 icons shown

To select a window to work in, highlight it by clicking on it.

These buttons always mean:

OK	**correct, do it**
Apply	**try it**
Cancel	**never mind**
Help	**explain this**

To press an onscreen button, click on it.



Move the pointer with the mouse.

Always use button 1 unless instructed otherwise.

click = press a mouse button briefly, then release it.

double-click = click twice in rapid succession.

drag = move something by pointing to it, then holding down a mouse button while moving the mouse.

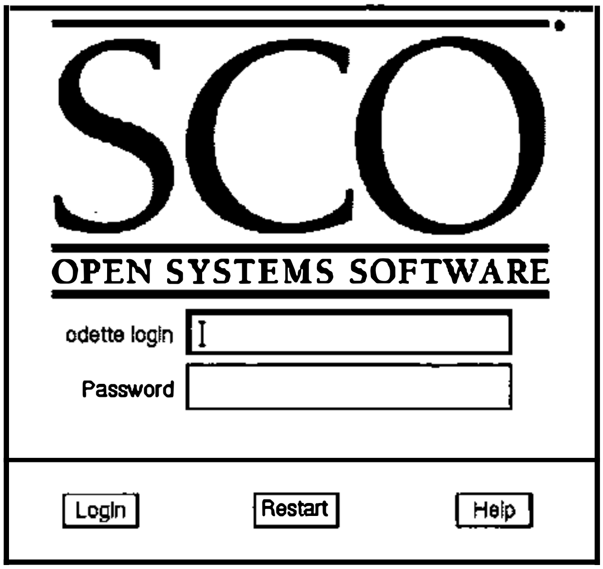
drop = deposit something in a new location by releasing the mouse button after dragging.

Left-handed people may want to exchange buttons 1 and 3.

Mouse

Use the Mouse control to do this or change mouse responsiveness.

Starting the Desktop



To start the Desktop, type your login name, press the ⟨Enter⟩ key, type your password, then press ⟨Enter⟩ again.

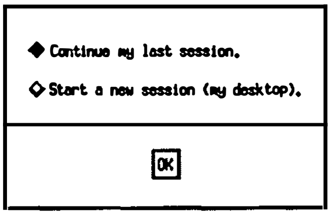
If the login line is not already highlighted as shown above, you must highlight (select) it before typing your login name. Use the mouse to move the arrow into the login line, then briefly press and release (click) the left mouse button.

If you do not have an account (login) name and password, see your system administrator or your UNIX system documentation.

To keep your password confidential, it is not shown on the screen as you type it. You should change your password regularly. Instructions for changing your password are in "Changing your password" (page 103).

If the login box shown above is not visible and your screen is blank, try pressing any key to wake up the screen saver. If your screen is still blank, make sure your terminal is turned on. If your screen is not blank, the absence of the login box means that the Desktop has been set up to start differently at your site. The most common alternative setup requires you to log in at a command line prompt, then type **startx**. See your system administrator for instructions.

Continuing your last session


◆ Continue my last session.

◇ Start a new session (my desktop).

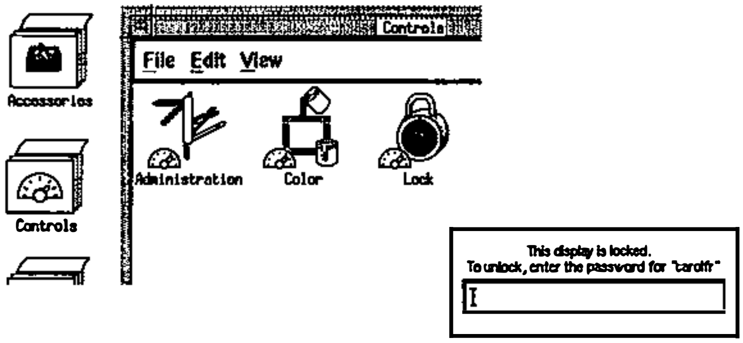
When the Desktop starts, the message shown above may appear. (It is not displayed the first time you start the Desktop.) To return to the Desktop configuration as you left it during your last session, click on the button next to "Continue my last session." To return to the default Desktop configuration, click on the button next to "Start a new session (my desktop)."

After you have made your choice, click on **OK** to continue. See Chapter 2 (page 11) for more about using the mouse to press onscreen buttons.

You can tell the software to always continue or start anew without asking you. See "Changing startup and shutdown" (page 54) for instructions.

If your screen goes blank after you have not been using your keyboard or mouse for a while, try pressing the ⟨Shift⟩ key. Screen saver programs blank your screen after it has been idle for a specified period. Pressing a key restores the display.

Locking your display



This display is locked.
To unlock, enter the password for "tarolff"

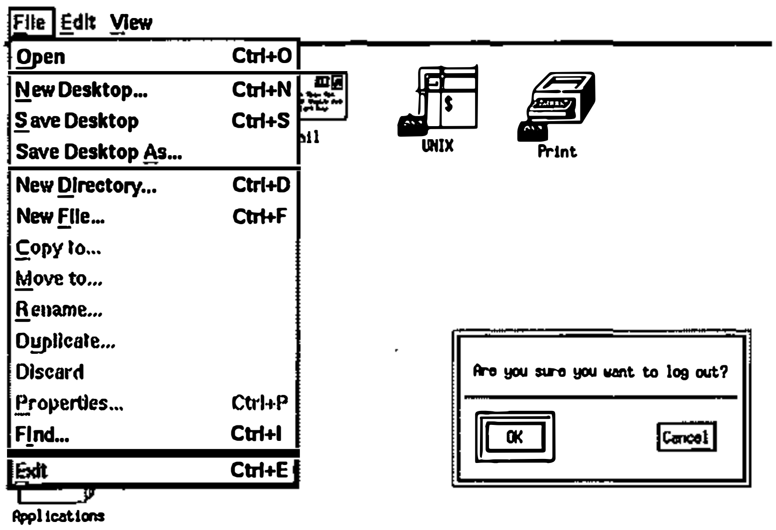
When you are away from your terminal, it is good practice to either log out or lock your display. This prevents anyone else from gaining access to your files or the system.

NOTE Save your current session before locking your display by selecting **Save Desktop** from the Desktop **File** menu. Your Desktop information is not saved if there is a power failure while your display is locked.

To lock your display:	Open the Controls window by double-clicking on the Controls icon. Double-click on the Lock icon. Your Desktop is replaced with the unlock box shown above.
To unlock your display:	Enter your login password (type it, then press ⟨Enter⟩) in the unlock box.

If you are running a program that interacts with the display, Lock temporarily interrupts it. When you unlock your display, the program continues from where it was interrupted.

Exiting the Desktop



Are you sure you want to log out?

To stop all your programs and leave the Desktop, click on the word **File** at the upper left corner of the Desktop. Then click on **Exit** in the **File** menu that opens.

When you are asked, click on **OK** to confirm that you want to log out. Any changes you have made to your Desktop are saved.

See Chapter 2 (page 11) for more about using menus or the mouse. See "Changing startup and shutdown" (page 54) for instructions on skipping the confirmation message.

Chapter 2

Clicking, moving, adjusting

SCO Open Desktop/SCO Open Server is a graphical environment. That means you can tell your computer what to do by pointing to graphic images (or *objects*) on the screen.

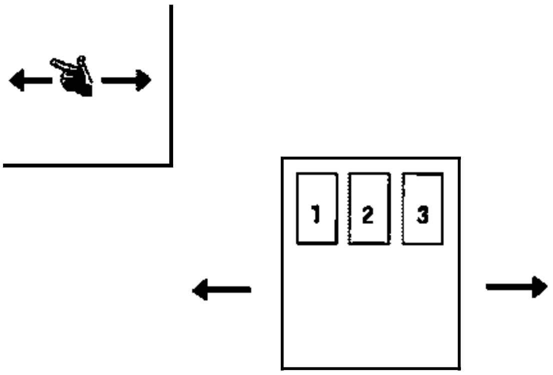
You must be familiar with four of these objects to work effectively on the Desktop: icons, windows, menus, and the pointer. In addition, you must know how to use a mouse to point to these objects and move them around.

Icons	are small pictures that represent programs, documents, data, or the containers in which they are kept
Windows	are work areas that open up on the Desktop. You look at and work with information through these windows.
Menus	are lists of things you can do inside a specific window, or outside windows on the Desktop itself
Pointer	shows you where you are on the screen. It changes shape to let you know what you can do there.

This chapter explains how to use these objects. If you are new to graphical environments, you may also want to read the *Tutorial.*

NOTE If you have worked with a mouse-based graphical environment that uses icons, windows, and menus, you can probably skip this chapter. Do review the quick-reference illustrations at the beginning of Chapter 1 if you have not already done so.

Using the mouse



Use the mouse to point to objects on the screen. When you move the mouse, the pointer (the pointing hand in the illustration) moves. When the pointer touches an object, you are pointing to it.

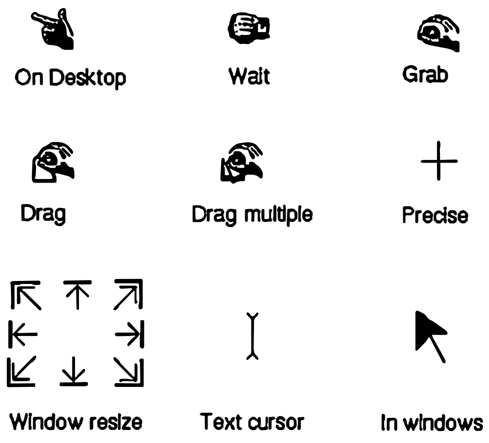
Use the buttons on the mouse to tell your computer what you want to do with the object. What you can do depends on which object you are pointing to, as explained throughout this chapter. But there are basically only three ways to use the mouse buttons:

Click	a button by briefly pressing and releasing it
Double-click	a button by clicking twice in rapid succession
Drag	(move) an object by pointing to it, then holding down a button while moving the mouse. Releasing the button *drops* the object in the new location.

For simplicity, we say "click on" to mean "point to an object then click the mouse button."

Always use mouse button 1 unless instructed otherwise. Mouse button 1 is usually the left button, as shown above. Left-handed people may prefer that the right button function as button 1. See "Adapting the mouse" (page 55) for instructions on swapping buttons 1 and 3.

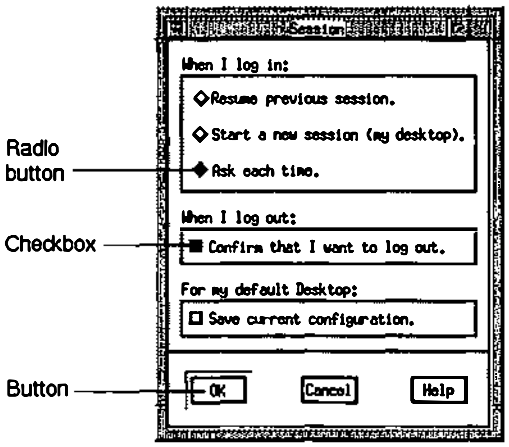
What different pointers mean



Although the pointer is usually a pointing hand, it sometimes changes into one of the above images. Each type of pointer tells you something about what you are allowed to do at the pointer's location at this time. The *wait* pointer, for example, tells you that the computer is busy doing what you asked, and that you must wait until it is ready for more instructions.

Individual programs may use pointers other than those shown above.

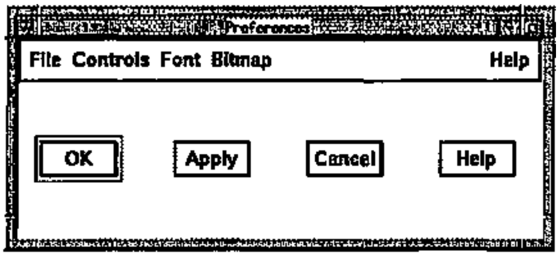
Pressing onscreen buttons


Radio button
Checkbox
Button
Session
When I log in:
◇ Resume previous session.
◇ Start a new session (my desktop).
◆ Ask each time.
When I log out:
■ Confirm that I want to log out.
For my default Desktop:
☐ Save current configuration.
OK
Cancel
Help

To press an onscreen button, click on it.

SCO Open Desktop/SCO Open Server programs make extensive use of onscreen "buttons." Like real pushbuttons, these graphical representations can be pressed to activate or select something. When pressed, onscreen buttons look like they have been pushed in. There are three major types of onscreen buttons:

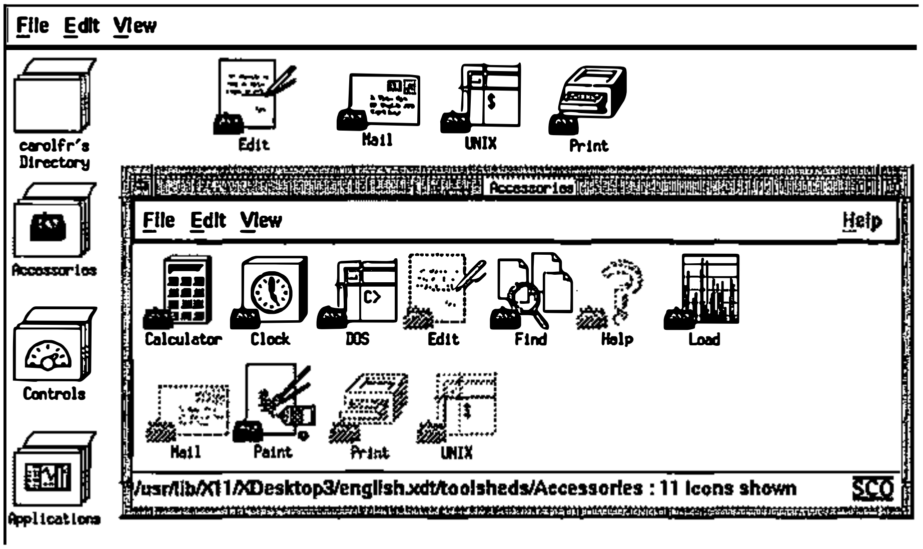
Buttons	are larger rectangles that initiate some action when pressed (like the **OK** or **Help** buttons).
Radio buttons	are small diamonds used to select exactly one of a group of options. Pressing a second radio button pops up the first (like a radio's preset station buttons).
Checkboxes	are small squares used to select one, more than one, or none of a group of options. Each checkbox individually toggles between selected and unselected.



You will encounter the four buttons shown above frequently, usually at the bottom of a window in which you can make choices or change things. They always initiate the same actions:

OK	accepts any changes you have made, closes the window, and initiates the appropriate action. In a message window, press **OK** to indicate you understand the message.
Apply	lets you try different settings. It shows you the effects of your changes, but leaves the window open. Your changes are not final until you press **OK**.
Cancel	undoes your changes and closes the window
Help	displays information about how to use the window

Controls, accessories, applications



The programs that run on the Desktop are organized into three categories:

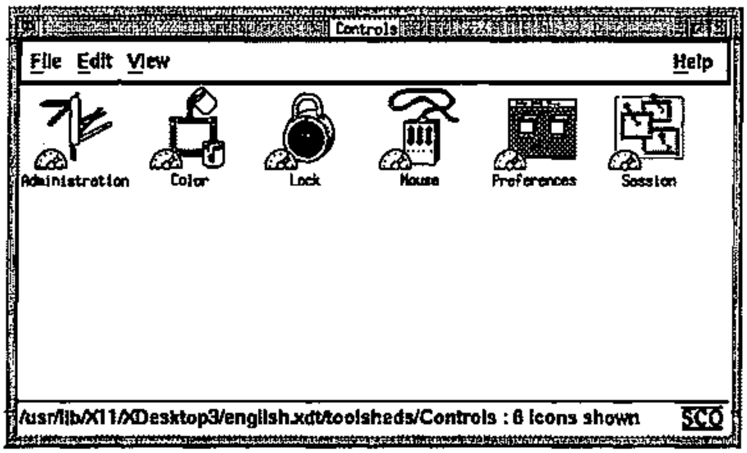
Controls	modify or control your Desktop environment, including colors, fonts, and start-up message
Accessories	are basic tools for working on the Desktop, including a calculator, editor, and clock
Applications	are major tools, such as spreadsheets, databases, and word processors

The controls and accessories are supplied with the Desktop. Applications are supplied separately by SCO or other companies.

The icons that represent the accessories are stored in the Accessories window, shown above. To open the Accessories window, double-click on the Accessories icon. Double-click on the Controls or Applications icons to open those windows. See "Opening and closing windows" (page 18).

Notice that some accessories have been moved out onto the Desktop, where they can be easily accessed even when the Accessories window is closed. (They leave an outlined placeholder behind to indicate that they still reside in the Accessories directory.) For more about moving program and other icons, see "Moving files and directories" (page 38).

Starting a program



/usr/lib/X11/XDesktop3/english.xdt/toolsheds/Controls : 6 icons shown

Start a control, accessory, or application program by double-clicking on its icon.

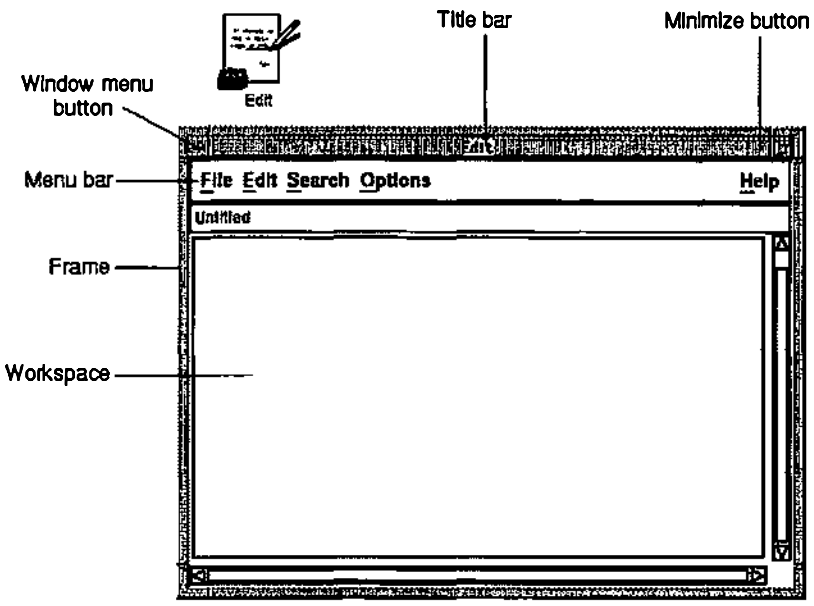
NOTE You may want to practice double-clicking until you get the timing right. You are double-clicking on (activating) a program icon correctly when the the pointer briefly changes to a wait pointer (a hand with a wristwatch) to indicate that the program is starting up. If the icon label highlights, you have only single-clicked on (selected) the icon.

You can also start some programs (such as Mail, Edit, and Print) by dropping a document (text file) icon on the program's icon. For example, dropping a document icon on Print prints the document.

For instructions on how to use a Desktop control or accessory, drop its icon on the Help icon. See Chapter 4 (page 43) for more about using Help. Instructions for using many controls and accessories are also included in this book (see the index).

Instructions for using applications are provided by the suppliers of the applications.

Opening and closing windows



Most of your work will be done inside one of the windows that open up on the Desktop. Some windows are opened by programs in response to something you do. Other windows you open directly by double-clicking on an icon to start a program or look at a document.

You can close a window in several ways:

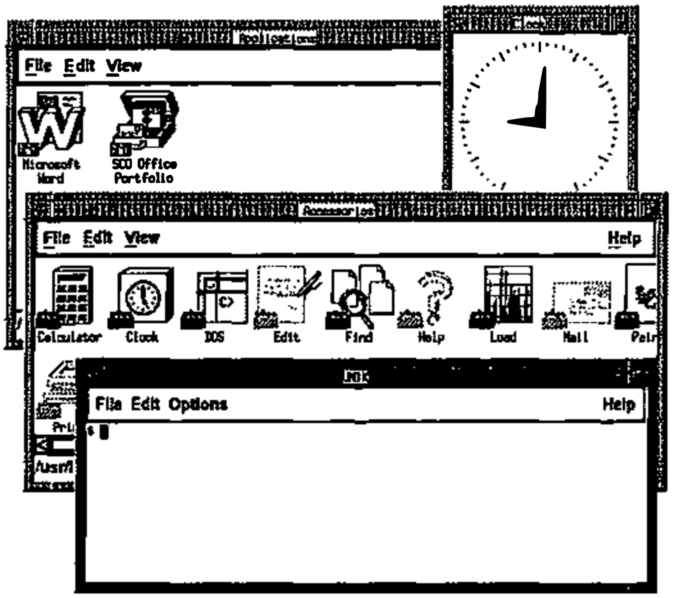
- Double-click on its **Window menu** button in the upper left corner of the window's frame.
- Select **Close** or **Exit** from its **File** menu (page 20).
- Select **Close** from its **Window** menu (page 21).

Selecting items from menus is explained in "Using menus" (page 20).

When you close a window in one of the above ways, all programs running in the window are stopped. To remove a window from the screen without stopping the programs running inside it, iconify the window instead of closing it:

- Click on the window's **Minimize** button to reduce the window to an icon.

Selecting a window

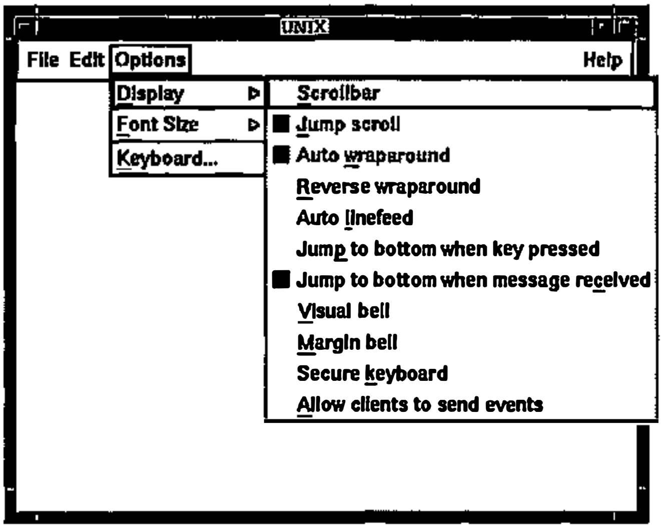


Select a window to work in by clicking on it. That window's frame changes color to identify it as the active window. If the window is partially covered by other windows, it jumps in front of them when selected. If the window you want to select is completely hidden by other windows, see "Moving windows" (page 23).

You can have many windows open at the same time. This lets you quickly move between programs or documents and compare work in one window with that in another. You can stack things up and move them around just as you would on an actual desk.

On your desk you can only write on one piece of paper at a time, even though you may move between papers rapidly and frequently. The same is true on the Desktop: you can only enter text, draw, or use controls in one window at a time. Things may be happening in other windows, but SCO Open Desktop/SCO Open Server needs to know to which window it should direct your input.

Using menus

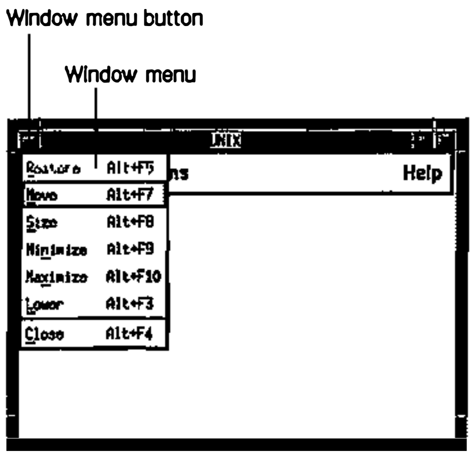


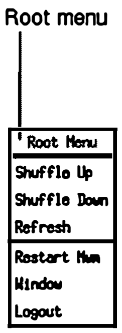
Most windows have a menu bar that lists the names of the menus available in that window. To open one of these menus, click on its name. To select an item from the open menu, click on that item.

Alternatively, you can *pull down* the menu by dragging the pointer down from its name in the menu bar until the item you want is highlighted. Release the mouse button to select the highlighted item.

To close the menu without selecting an item, move the pointer off the menu and click (or release) the mouse button.

If the item you select requires that you make additional choices, a submenu or window appears. Use these like any other menu or window.





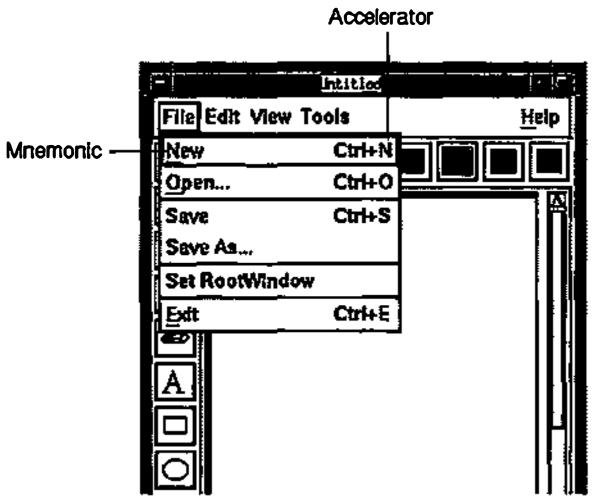
You will find that certain types of menus are common to most windows. **File** menus list actions that affect the document (file) or program as a whole, such as save, print, and exit. **Edit** menus list actions that affect the text or data in the window, such as cut and paste, copy, and delete. **View** menus list actions that change the appearance, amount, or arrangement of the window's contents. Use **Option** menus to customize the program running in the window. **Help** menus provide access to information about using the window.

In addition to its menu bar menus, every window has a **Window** menu you can use to move, resize, or close the window. Open this menu by clicking on the **Window** menu button in the upper left corner of the window's frame.

A similar menu, called a **Root** menu, is available from the Desktop. Open it by pointing to the edge of the Desktop and holding down the mouse button. Drag the pointer to highlight the item you want, then release the button.

NOTE Sometimes network line noise or similar problems can garble your display. If this occurs, select **Refresh** from the **Root** menu.

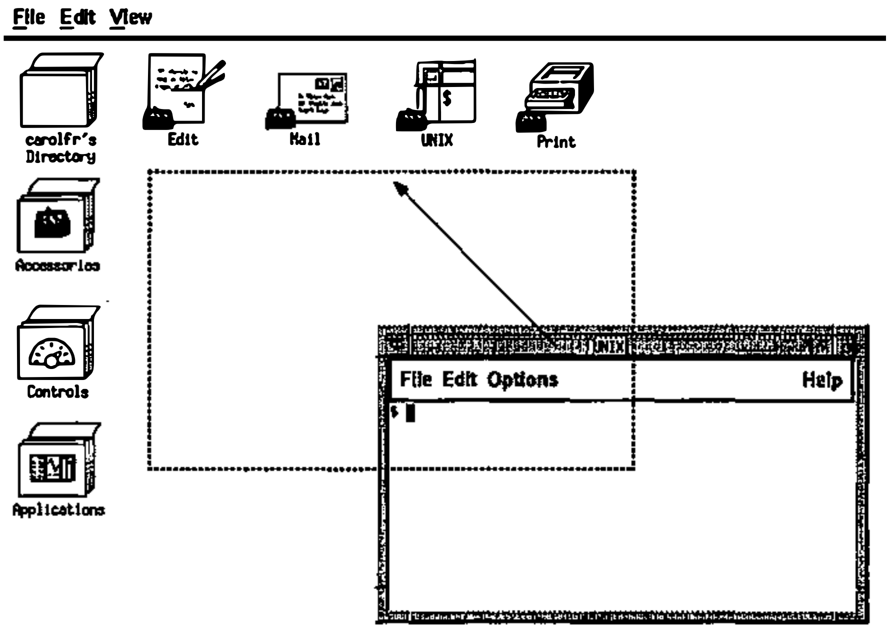
Menu shortcuts



When a menu is open, you can select an item from it without using the mouse by typing the underlined letter (*mnemonic*) in the item's name. For example, you could type "O" to select **Open** from the above menu.

Some frequently used menu items can be selected from the keyboard without opening the menu. Such items are identified by a combination of keys (⟨Ctrl⟩**E**, for example) to the right of the item in the menu. Press those keys simultaneously to select the item. The key combinations used to select items from unopened menus are called *accelerators*.

Moving windows

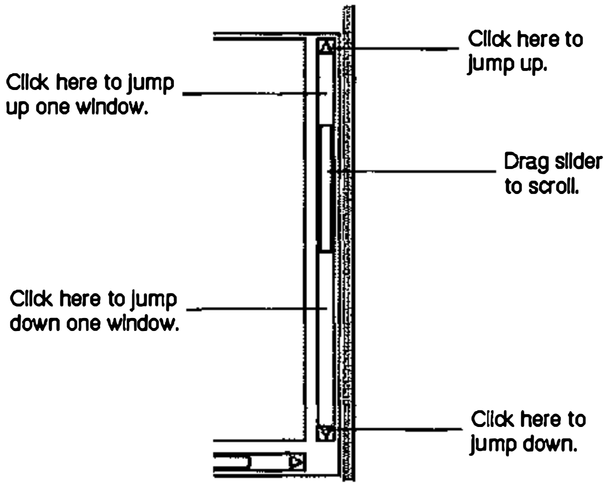


You can move windows around as you would papers on a desk. To move a window, drag it by its title bar.

If the window you want is hidden under a stack of windows, repeatedly select **Shuffle Up** from the **Root** menu (page 21) until that window appears at the top of the stack.

To move a window from the top to the bottom of a stack, select **Shuffle Down** from the **Root** menu. Or, select **Lower** from the window's **Window** menu (page 21).

Scrolling


Click here to jump up one window.
Click here to jump down one window.
Click here to jump up.
Drag slider to scroll.
Click here to jump down.

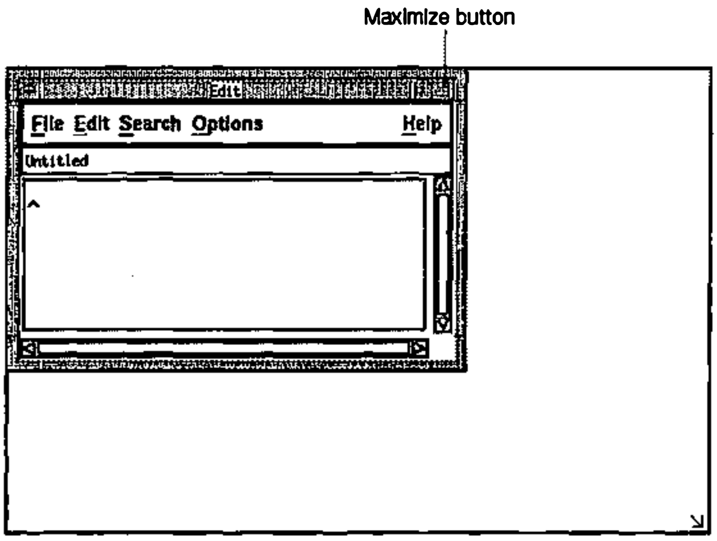
When there is more information to display than can fit in a window, use the scroll bars to see more.

- To scroll smoothly through the window's contents, drag the scroll bar slider.
- To move in short jumps toward the top or bottom (or right or left edge) of the window's contents, click on the arrow at either end of the scroll bar.
- To jump by one full window, click on the scroll bar between the slider and one of the arrows.

The position of the slider in the scroll bar indicates how much of the window's contents lay to either side of the portion currently displayed in the window.

You can also make the window bigger to display more of its contents. See "Resizing windows" (page 25) for instructions.

Resizing windows



Change the size of a window by dragging a side or corner of its frame.

NOTE If you change the size of a UNIX window, you must enter **resize** at the UNIX prompt before entering any other commands.

To fill the whole screen with a window, click on the **Maximize** button at the upper right corner of the window's frame. To restore a maximized window to its previous size, click on the **Maximize** button again.

Chapter 3

Working with text

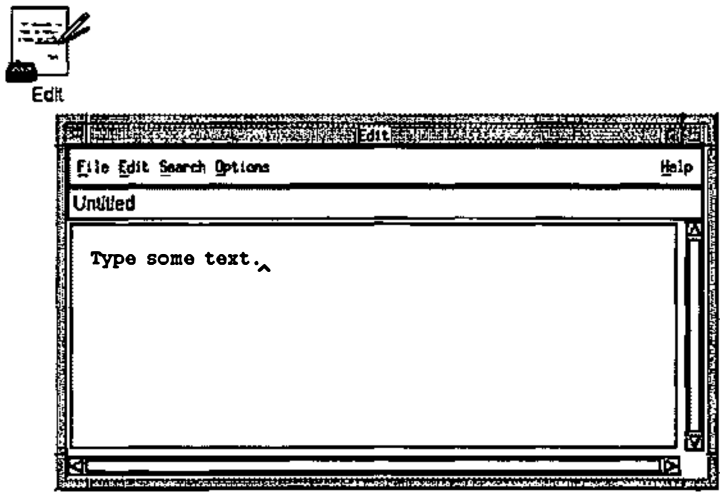
SCO Open Desktop/SCO Open Server provides a number of tools to help you write, edit, and organize text. These include programs for editing, printing, filing, and finding text.

Each letter, memo, report, or other document you work with is stored in a *file.* In fact, all of your work on the Desktop is organized in files. A file is just a named collection of information. It can contain text, commands, a program, or information used by a specific program, such as database, spreadsheet, or drawing software.

Files are organized in *directories.* A directory is a place to put files, like a file box or drawer. And like the label on a file drawer, each directory has a name. Unlike file drawers, you can put directories inside other directories. This gives you the ability to organize your work in very useful ways, as we will see later in this chapter (page 35). Before we can organize files, however, we must first create them.

NOTE Although this chapter focuses on working with text files, much of the information about using files and directories applies to other kinds of files as well.

Creating files



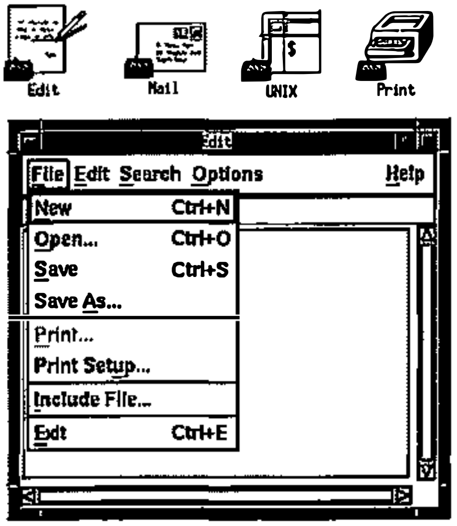
Type some text.

To create a text file, double-click on the Edit icon. When the Edit window opens, move the pointer inside the window and click. Type some text.

You can also create text or other files from inside appropriate accessories (Edit, Paint, Mail) or applications (word processors, spreadsheets). Generally, you do this by clicking on the program's icon or selecting **New File** from a **File** menu. For specific instructions on creating files inside an application, see the application's manual. For detailed instructions on creating files inside an accessory or directory window, use the window's **Help** menu (page 43).

Another way to create a new file is by copying an existing one. See "Copying files or directories" (page 39).

Saving files



After you have entered some text, select **Save** from the **File** menu. You are asked to name the file when you save it for the first time.

Choose a name that helps you to remember what is in the file (for example, *letter.Susan* for a letter to Susan). Keep the name short so it is easy to type. Do not use blank spaces or these characters in filenames:

! ; * / \ $ & < > ( ) | { } ~ [ ] ? - " ` '

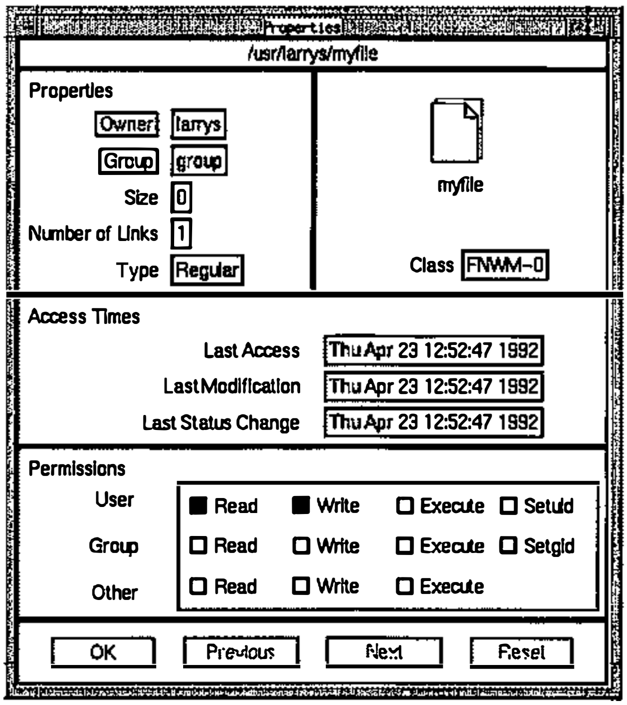
Uppercase letters are different from lowercase letters. For example, *letter.susan* and *letter.Susan* are different filenames.

NOTE It is a good idea to save your work frequently, so you do not lose it if your system goes down due to a power failure or other problem.

To save your work to a different file, select **Save As** from the **File** menu. Edit asks you for the name of the file you want to save your work in.

Select **Exit** from the Edit window's **File** menu. If you have changed the text in the file since the last time you saved it, Edit asks you if you want to save those changes when you exit.

Controlling access to files



/usr/larrys/myfile

Properties

Owner	larrys
Group	group
Size	0
Number of Links	1
Type	Regular

myfile

Class FNWM-0

Access Times

Last Access	Thu Apr 23 12:52:47 1992
Last Modification	Thu Apr 23 12:52:47 1992
Last Status Change	Thu Apr 23 12:52:47 1992

Permissions

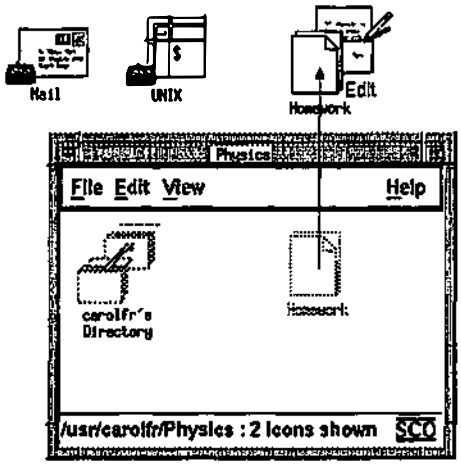
User	■ Read	■ Write	□ Execute	□ Setuid
Group	□ Read	□ Write	□ Execute	□ Setgid
Other	□ Read	□ Write	□ Execute	

SCO Open Desktop/SCO Open Server files have a built-in security feature that lets you control who gets to look at or change your files. You can also control who can run (execute) your program files.

These permissions (also called properties or attributes) are set separately for yourself as the creator or owner of the file, for your work group (ask your system administrator how work groups are set up at your site), and for all others with access to your computer. For example, you could decide that you could look at (read) and change (write) a file, that your group could read it but not write to it, and that others could not even read it.

You can check the permissions for a file by clicking on its icon, then selecting **Properties** from the **File** menu of the window containing the icon. The Properties window also gives you information about the size and type of the file, as well as the name of its owner and the time it was last changed. If you own the file, you can change the permissions by clicking on the appropriate checkboxes in the Properties window.

Viewing files



To view the contents of a text file, double-click on the file's icon. Or, drag the file's icon onto the Edit icon and drop it. Either method displays the file inside the Edit window.

When you are done looking at the file, select **Exit** from the Edit window's **File** menu.

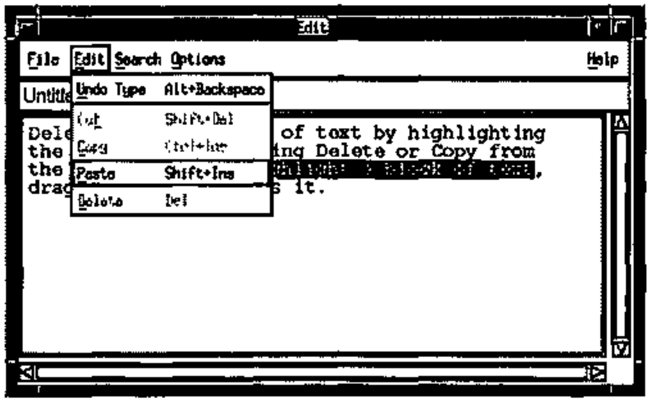
Hidden files

Certain file icons are normally hidden from view. These represent *dot files,* so called because their names start with a period (or "dot"). By convention, files that contain configuration information for the Desktop, other programs, or your computer environment have names beginning with a period. Configuration files are hidden to eliminate clutter in your directory windows (page 34), because you rarely need to view or modify them.

When you do want to see what dot files reside in a directory, select **Display dot files** from the window's **View** menu, then select **Show** from the submenu. To hide the dot files again, select **Display dot files** from the window's **View** menu, then select **Hide** from the submenu.

Editing files

Edit



To edit an existing file, double-click on its icon or drop its icon on the Edit icon. Or, double-click on the Edit icon, then select **Open** from the Edit window's **File** menu; you are asked for the name of the file you want to edit.

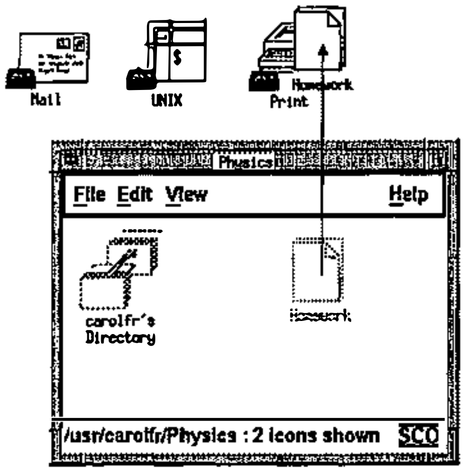
To enter text in a file, the pointer must be in the Edit window. Move around in the file with the mouse, scroll bars, or arrow keys. To move the cursor (the symbol that shows where text you type will be inserted), point to a new position and click.

NOTE It is a good idea to save your work (page 29) frequently so you do not lose it if your system goes down due to a power failure or other problem.

Delete or copy blocks of text by highlighting the text, then selecting **Delete** or **Copy** from the **Edit** menu. To highlight a block of text, drag the cursor across it. You can insert text you have just deleted by moving the cursor to where you want to insert the text, then selecting **Paste** from the **Edit** menu. Use the **Search** menu to find text and to replace found text with different text. For additional information about editing files, use the Edit window's **Help** menu (page 43).

When you are finished editing the file, select **Exit** from the Edit window's **File** menu.

Printing files



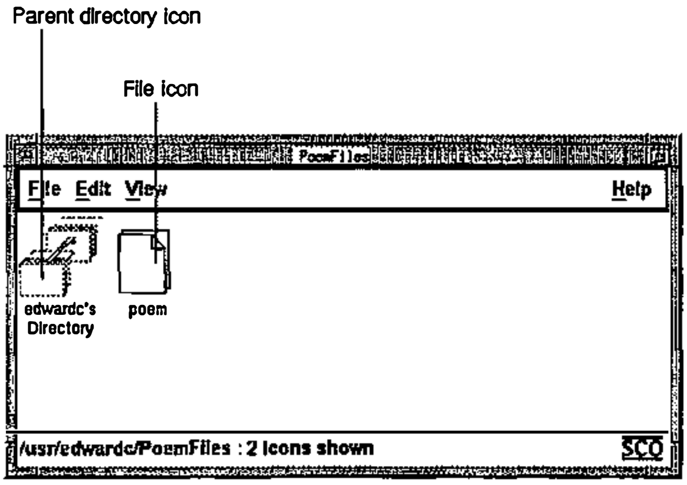
Print a text file by dropping its icon on the Print icon. The file is printed on the default printer. For information on how to set up or change the default printer, see the *System Administrator's Guide.*

In some applications, you can print files by selecting **Print** from the application's File menu. See the application's manual for instructions.

Instructions for printing a mail message are in Chapter 5 (page 49).

Options such as multiple copies and printer selection are available when you print a file from within the UNIX window using the **lp** command. See Chapter 10 (page 77) for instructions.

Viewing directory contents



/usr/edwardc/PoemFiles : 2 icons shown

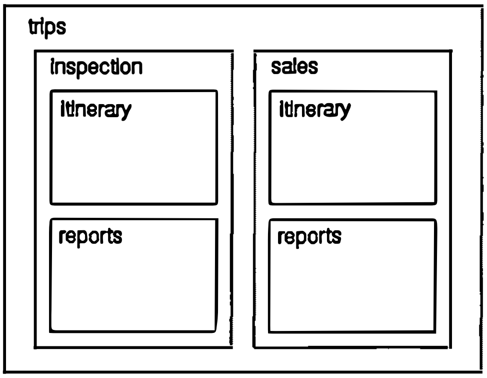
A directory is like a file box; it is a place to put files. Like files, directories have names and are represented by icons. However, instead of containing text or data, directories contain files and other directories.

To see what is in a directory, double-click on its icon. The icon opens into a directory window that contains file and directory icons. You can move, resize, close, or scroll in a directory window just like you would in any other window (see Chapter 2). You can also open a directory by selecting **Open** from the Desktop **File** menu. To display the icons for hidden files (page 31), select **Display dot files** from the directory window's **View** menu.

If you double-click on a directory icon inside a directory window, a second directory window opens. You can save time and open fewer windows by instead double-clicking on the directory icon with mouse button 2 (on a two-button mouse, double-click with both buttons simultaneously). This replaces the contents of the open directory window with those of the icon's directory. Double-click on the parent directory icon with mouse button 2 to redisplay the first directory.

To print a list of the directory's contents, drop the directory's icon on the Print icon.

Organizing your files



Every file resides in a directory. The *current directory* is the one in which you are currently working. When you are working on the Desktop, your *home directory* (the one with the same name you use to log in) is the current directory. Files reside in the directory that was current when they were created unless you specify that they belong in another directory.

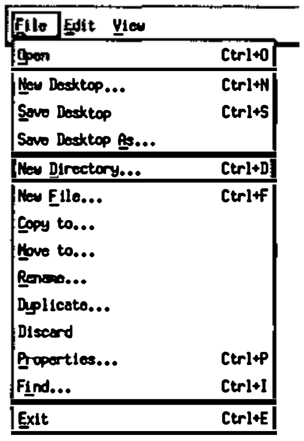
Use directories to organize your files so they are easy to find and work with. For example, you could put all your trip reports in a directory named *trips.* If you take a lot of trips, you may want to divide the reports into *inspection* and *sales* directories. The contents of each of these directories might be further divided into *itinerary* and *report* subdirectories. (*Subdirectory* is another word for a directory contained in another directory.)

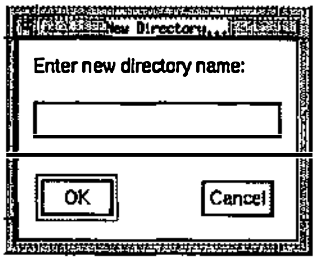
The location of a file is specified by a *pathname,* which consists of the file's name and the path to it. The path is a list of the directories have to be opened to find the file. In the example above, the pathname for a sales report named *April.92* would be */usr/nina/trips/sales/reports/April.92* (where */usr/nina* is the name of the home directory). Slashes separate directory and file names in paths. See "Specifying directory names" (page 73) for more about paths.

Some applications (especially databases and spreadsheets) expect that their data files be organized in specific directories and subdirectories. The applications' manuals will explain any such requirements.

NOTE Each file and subdirectory in a directory must have a unique name (so you and the software can tell them apart). Files and directories can have identical names if they reside in different directories.

Creating directories





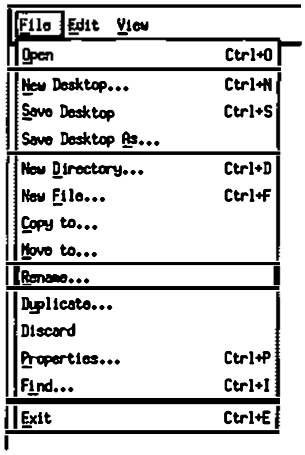
To create a new directory, select **New Directory** from the **File** menu of a directory window or the Desktop. When asked, enter the name of the new directory.

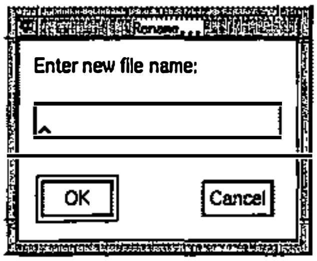
To create a new directory anywhere other than in the current directory, enter the pathname (page 35), beginning with the directory name after the home directory. For example, enter *trips/sales/reports/drafts* to create a new subdirectory named *drafts* inside the *reports* subdirectory of the *sales* subdirectory of the *trips* directory under the */usr/nina* home directory.

The new directory icon appears in the directory window in which you created it. If you create a directory on the Desktop, the icon appears on the Desktop, but it actually resides in your home directory.

Directories have permission settings, just like files. You must have write permission in a directory before you can create a new directory there. You must have execute permission for a directory before you can make it the current directory. You can check and change directory permissions in the same way you would for files. See "Controlling access to files" (page 30) for instructions.

Renaming files and directories



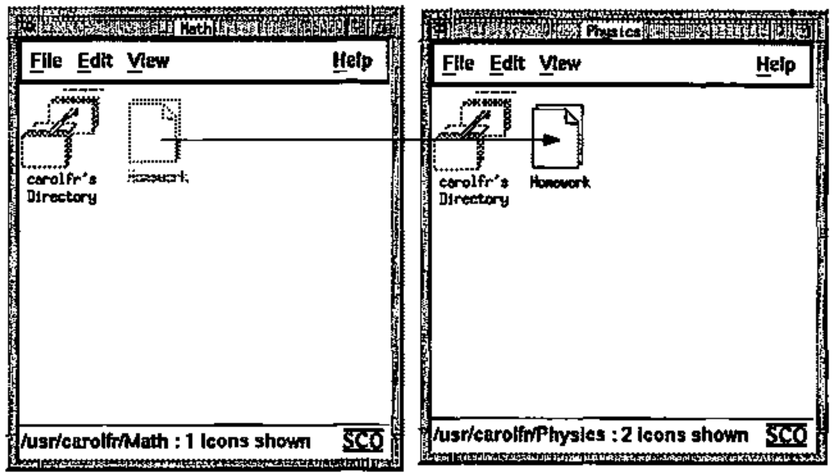


To change the name of a file or directory, click on its icon, then select **Rename** from the **File** menu of the directory window (or Desktop) that contains it. When asked, type the new name. Click on **OK**.

You must have write permission for a file or directory to rename it. See "Controlling access to files" (page 30) for more about permissions.

Files can have more than one name. See "Copying files or directories" (page 39) for more about multiple names.

Moving files and directories

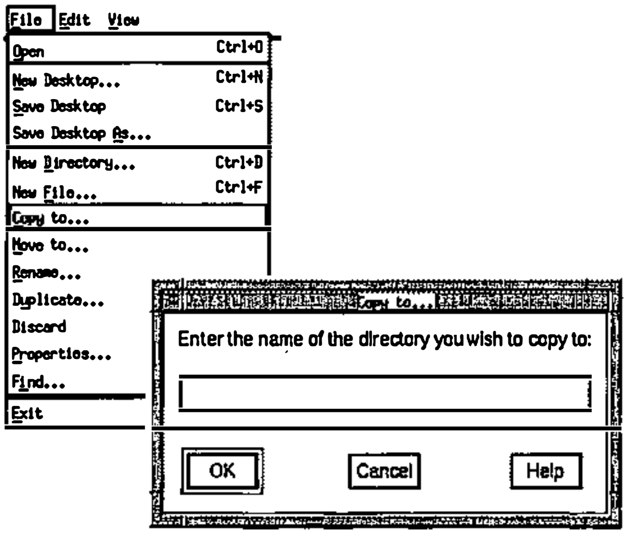


To move a file or directory from one directory to another, drag its icon into the destination directory's window. If the directory window is not open, drop the icon on the directory's icon. Move files or directories from the Desktop to a directory window in the same way.

Another method for moving files and directories uses the Desktop **File** menu. Click on a file or directory icon, then select **Move to** from the Desktop **File** menu. When asked, enter the name of the destination directory, then click on **OK**. If the destination directory is contained inside other directories, you must list those directories as well, each separated by slashes. For example, to move the file *April* into the *itinerary* directory, which is in the *sales* subdirectory of the *trips* directory, enter *trips/sales/itinerary*. This is called a path because it tells how to get from the Desktop to the directory. The directory's pathname is shown at the bottom of each directory window. See "Specifying directory names" (page 73) for more about paths.

NOTE You must have write permission for the destination directory, and all the directories in the path must already exist. You may not move your home directory or the trash directory. See "Controlling access to files" (page 30) for more about permissions.

Copying files or directories



Enter the name of the directory you wish to copy to:

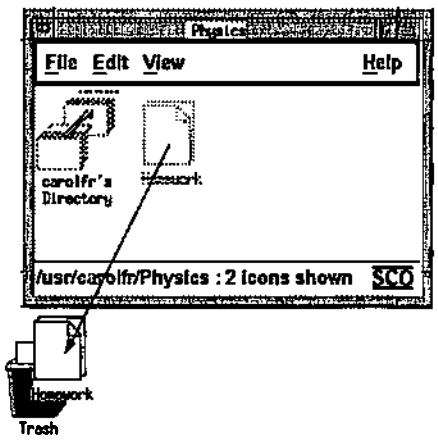
Making a copy of a file or directory is just like moving it, except you hold down the ⟨Ctrl⟩ key while you drag the icon to its new location.

You can also copy files and directories by selecting **Copy to** from the **File** menu.

Because each file and subdirectory in a directory must have a unique name, you cannot make a copy of a file or directory within the same directory by dragging and dropping an icon. To copy within a directory, click on the icon, then select **Duplicate** from the **File** menu. When asked, enter a name (page 29) for the copy you are creating, then click on **OK**.

Another way to have a file available in more than one directory is to give it more than one name. These additional names are called *links* because they are linked to a single copy of the file. Links differ from copies in that changes to one also changes the others. See "Linking files" (page 79) for more about links.

Trashing files and directories



To delete a file or directory, drag its icon onto the Trash icon and drop it.

Or, click on a file or directory icon, then select **Discard** from the **File** menu.

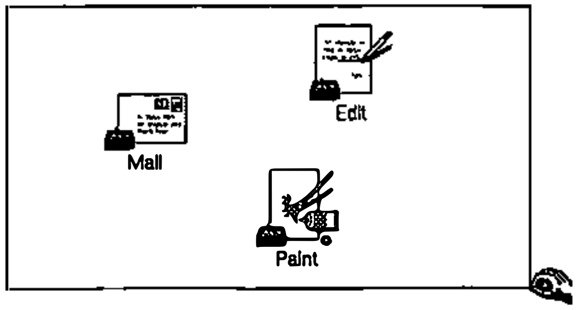
NOTE You must have write permission for a directory to trash files or directories in it. You may not throw away your home directory or the trash directory.

When you throw files and directories into the trash, they disappear from the screen, but are kept in a trash directory. This lets you dig them back out of the trash if you decide you did not want to throw them away after all.

T-· see what is in the trash, double-click on the trash icon to open the Trash window. You should periodically empty the trash so your disk does not fill up with files you no longer want. Once a file has been emptied from the trash, it cannot be recovered.

Use the trash window's **File** menu to recover files or empty the trash. To get detailed instructions, use the trash window's **Help** menu (page 43).

Working with multiple files or directories



You can move, copy, print, or trash more than one file or directory at a time. Hold down the ⟨Ctrl⟩ key while clicking on the icon for each file or directory you want to work with. This selects a group of icons. Dragging any one of the selected icons drags the whole group. Dropping that icon on, for example, the trash icon trashes the whole group. Selecting a menu item (such as **Copy to**) causes its action to be performed on the entire group of selected icons.

You can also select a group of icons by pointing to an empty spot near the icons and dragging. A selection box like the one shown above appears. Drag the box until it encompasses the icons, then release the mouse button.

To unselect one of the icons in a selected group, click on it while holding down the ⟨Ctrl⟩ key. To unselect the whole group, click on an empty spot.

Chapter 4

Getting help

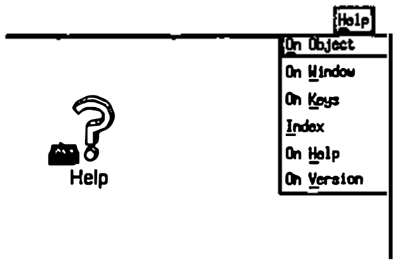
You can get information about, or instructions for using controls, accessories, files, directories, menus, windows, and other SCO Open Desktop/SCO Open Server components and features right on the screen. Many topics are covered in greater detail in the *online documentation* than here in this book.

You can access this information by pointing to the object you want information about, or by looking up a topic in an online index. In the index, you can click on a topic to display. While reading, you can display related illustrations and definitions of terms, or jump to related topics with the click of a mouse button.

In addition to explaining how to use the Desktop, the Help accessory provides access to online books on other SCO Open Desktop/SCO Open Server subjects, including *Features and Limitations* and *User and Administrator Manual Pages.*

The next two pages tell you how to display help information. The rest of the chapter explains how to move around within that information.

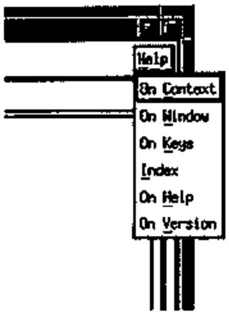
Getting help on the Desktop or in directory windows



For information about an icon, drop it on the Help icon. Or, click on the icon, then select **On Object** from the Desktop **Help** menu.

For information about a topic, double-click on that topic in the Help index. To display the index, select **On Index** from the Desktop **Help** menu.

Getting help inside other windows

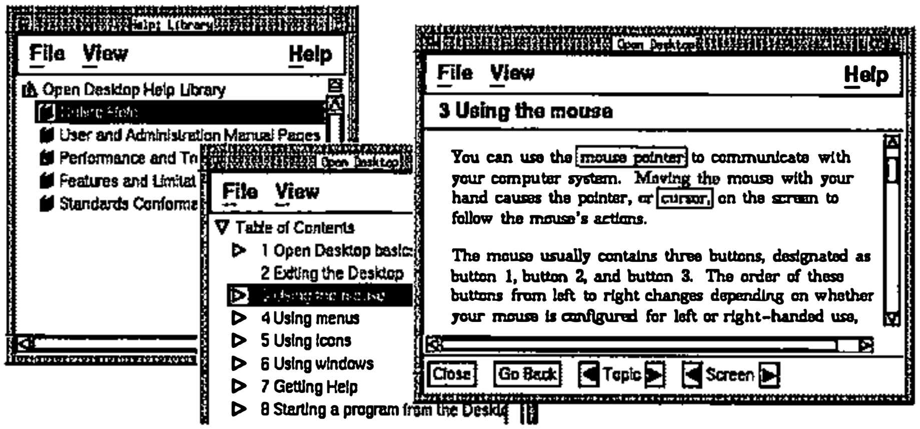


To get general information on the program running inside a window, select the appropriate item from the window's **Help** menu.

For information about something in the window, point to it and press the ⟨F1⟩ key. Or, select **On Context** from the window's **Help** menu (the pointer changes to the question-mark help pointer), then click on the part of the window you want to know about.

NOTE You cannot use the ⟨F1⟩ key to get context-sensitive help in the UNIX or DOS windows, but you can display the traditional manual pages for UNIX commands. See "More information about commands" (page 65).

Reading help



3 Using the mouse

You can use the mouse pointer to communicate with your computer system. Moving the mouse with your hand causes the pointer, or cursor, on the screen to follow the mouse's actions.

The mouse usually contains three buttons, designated as button 1, button 2, and button 3. The order of these buttons from left to right changes depending on whether your mouse is configured for left or right-handed use,

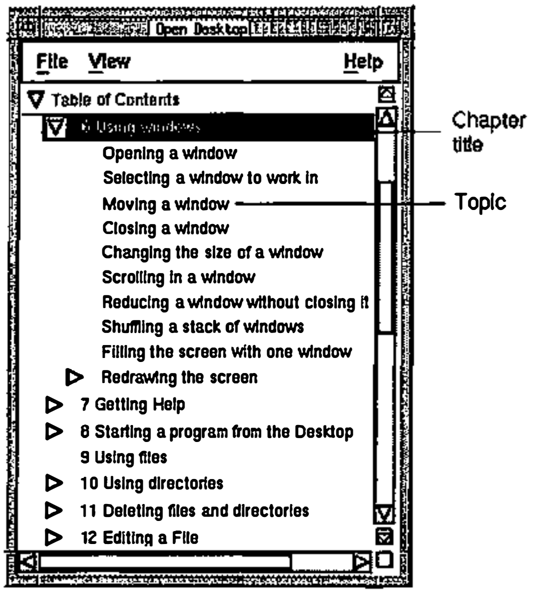
Information is displayed in the Help Topic window. Use the arrow buttons at the bottom of the Topic window to jump forward or back by a screen (window full) or topic. Or, use the scroll bars.

Boxes around words indicate point-and-jump cross-references (also called *hyperlinks* or *hot spots*). Click on a boxed word to display an illustration, table, definition, or related topic. Select **Hotspot** from the Topic window's **View** menu to hide the boxes (when hidden, they appear only if you move the pointer over them).

To see a table of contents or index, select **Open Table of Contents** from the Topic window's **File** menu.

To switch to another book, double-click on the title of the book in the Help Library window. If the Library window is not open, double-click on the Help Library icon, usually located at the bottom right corner of the Desktop. If the Help Library icon is not visible, double-click on the Help icon (the Help Library icon only appears after the first call for Help in an Open Desktop session).

Using the table of contents and index



▽ Table of Contents
▽ 6 Using windows
Opening a window
Selecting a window to work in
Moving a window
Closing a window
Changing the size of a window
Scrolling in a window
Reducing a window without closing it
Shuffling a stack of windows
Filling the screen with one window
▷ Redrawing the screen
▷ 7 Getting Help
▷ 8 Starting a program from the Desktop
9 Using files
▷ 10 Using directories
▷ 11 Deleting files and directories
▷ 12 Editing a File

When you open an online book, the Help Book window displays its table of contents, index, and table of figures and tables (if any). The title of the online book is shown at the top of the Book window.

To see the contents of a chapter (or subentries for an index entry), double-click on the triangle next to the chapter title (or index entry). Click on it again to hide the subtopics. The triangle points left when the subtopics are hidden (collapsed), down when they are visible (expanded). Use the Book window's **View** menu to expand or collapse all topics at once.

Double-click on a topic to display it.

Chapter 5

Sending and reading mail

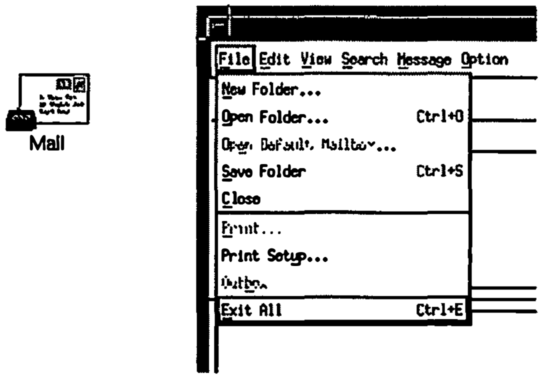
The SCO Open Desktop/SCO Open Server Mail accessory provides convenient and easy access to electronic mail. This chapter covers the basics: reading, sending, and replying to mail messages.

In addition, with Mail you can:

- print copies of your mail
- save file copies of all mail that you send
- organize your mail in folders by subject, date, sender, or other convenient category
- delay delivery of messages you send for a specified time
- hold a message you are composing until you are ready to finish and send it
- include your real name with your login on mail you send
- end your mail with a personalized *signature* message

Select these features from Mail menus. For detailed instructions, use the Mail **Help** menu.

Opening and closing Mail



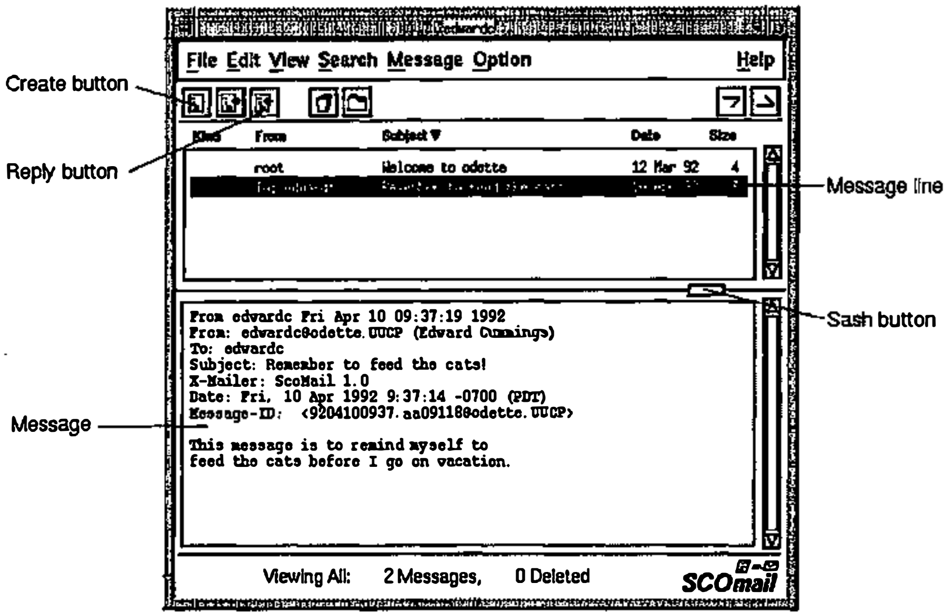
To open th e Mail accessory window, double-click on the Mail icon.

The Mailbox window shown on the next page opens to show you a list of mail waiting to be read.

A message at the bottom of the Mailbox notifies you when new mail arrives. If you reduce the Mail accessory to an icon, the icon changes to a letter in an envelope to indicate new mail has arrived.

To exit Mail, select **Exit All** from the Mailbox window's **File** menu. You are asked if you want to save any changes (you may, for example, have deleted mail once you read it). To discard all such changes since the last save, exit without saving.

Reading and replying to mail


Create button
Reply button
Message line
Sash button
Message
File Edit View Search Message Option Help
Kind From Subject Date Size
SCOmail

```
root          Welcome to odette          12 Mar 92   4
```

```
From edvardc Fri Apr 10 09:37:19 1992
From: edvardc@odette.UUCP (Edward Cummings)
To: edvardc
Subject: Remember to feed the cats!
X-Mailer: ScoMail 1.0
Date: Fri, 10 Apr 1992 9:37:14 -0700 (PDT)
Message-ID:  <9204100937.aa09118@odette.UUCP>

This message is to remind myself to
feed the cats before I go on vacation.
```

Viewing All: 2 Messages, 0 Deleted

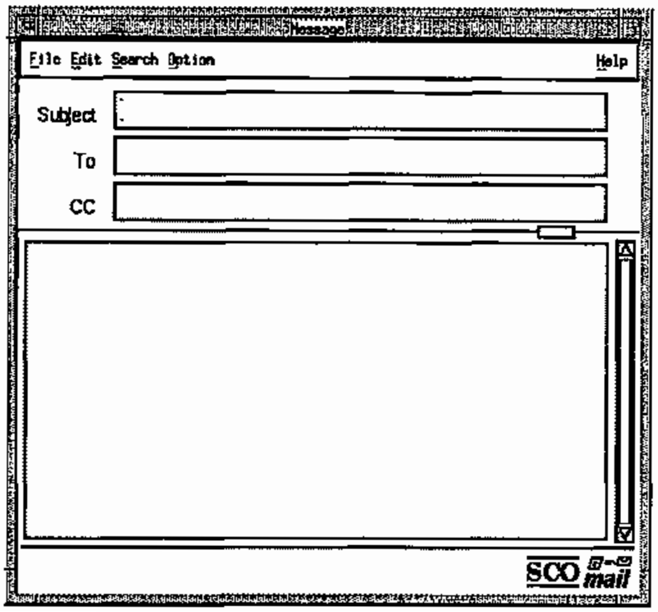
To read a mail message, click on the corresponding message line.

The work area of the Mailbox window is divided into two parts. The message you select from the list in the upper part is displayed in the lower part. Drag the sash button on the divider up or down to change the amount of window devoted to either part.

To sort the messages by kind, from, subject, date, or size, click on that column title at the top of the message list.

To reply to a message, click on the **Reply** button. On the panel that pops up, click on buttons to indicate whether you want to reply to the sender only or also to everyone else who received a copy of the message, whether you want to include the message in your reply, and if you do, whether you want to indent the message to set it off from your reply. Click on **Edit** to open the Message window. Compose your message as described on the next page. When you are finished, select **Deliver** from the **File** menu to send your reply.

Sending mail



To send a mail message, click on the **Create** button in the Mailbox window. This opens the Message window shown above.

Enter the subject of the message and the login names of the people you want to send the message or copies ("CC") of the message to. Click on each box before typing in it. You can enter more than one name or alias (page 237) in either the "To" or "CC" boxes (be sure to separate each name or alias with a space).

Type the message in the main work area. You can enter and edit text just like in the Edit window (page 32). To include a text file or another message in your message, select **Include File...** from the **File** menu, then click on a file from the list that is displayed. Outside the Mail window, you can mail a text file by dropping its icon on the Mail icon (you will asked where you want to send the file).

When you are finished, select **Deliver** from the **File** menu to send the mail.

Chapter 6

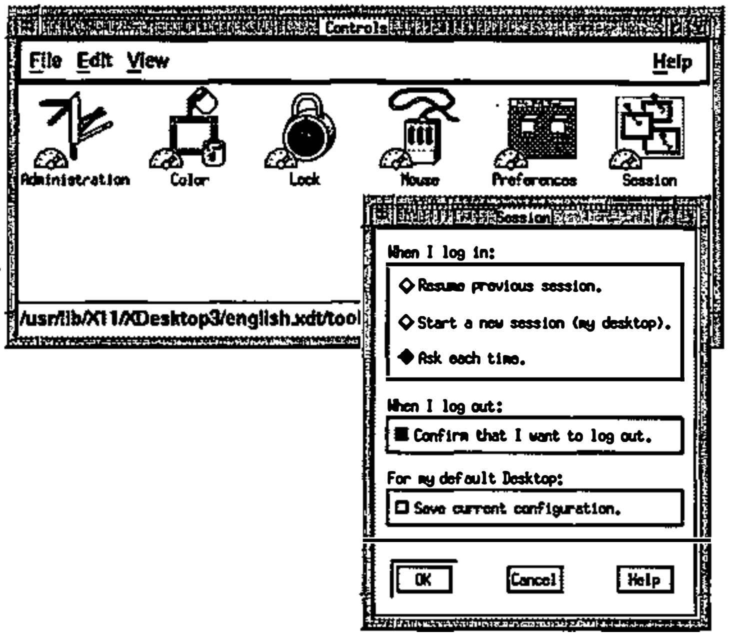
Customizing your Desktop

Your Desktop is your personal work environment. You can customize its organization and appearance to match your work style and make performance of your daily tasks as easy and efficient as possible.

The Controls window (opened by double-clicking on the Controls icon) contains a number of useful tools for modifying colors, mouse configuration, fonts, background patterns, startup and shutdown options, and other Desktop behavior.

For information on customizing Desktop in ways not covered in this chapter, refer to the online Help documentation (see Chapter 4) or the *Graphical Environment Administrator's Guide.*

Changing startup and shutdown



When I log in:

◇ Resume previous session.

◇ Start a new session (my desktop).

◆ Ask each time.

When I log out:

■ Confirm that I want to log out.

For my default Desktop:

☐ Save current configuration.

Double-click on the Controls icon to open the Controls window, then double-click on the Session icon to open the window shown above.

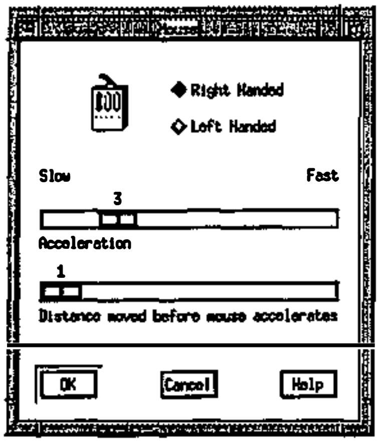
To always continue your last session where you left off, click on the button next to "Resume previous session." To start each session with the default Desktop configuration, select "Start a new session (my desktop)." To be asked which you prefer at the start of each session, select "Ask each time."

To avoid accidentally logging out, select "Confirm that I want to log out."

To define a new default configuration for the Desktop, arrange the icons and windows as you would like them to come up at the start of a session, then select "Save current configuration."

Click on **OK** when you have made your choices.

Adapting the mouse



Double-click on the Controls icon to open the Controls window, then double-click on the Mouse icon to open the window shown above.

To swap mouse buttons 1 and 3, select "Right Handed" or "Left Handed".

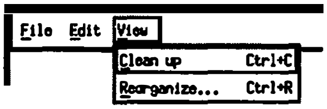
NOTE All adjustments made in the Mouse control are effective immediately. This lets you try your changes as you make them, but can be confusing if you select Left Handed and do not realize that you must now click the *right* mouse button instead of the left.

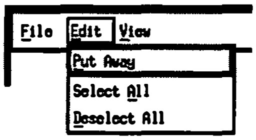
If you click a mouse button and nothing happens, the mouse may be configured for use with the other hand.

Drag the **Acceleration** slider to set the speed to which the pointer accelerates after being moved a short distance. The closer to 10, the faster the pointer moves. This lets you make short, precise movements, but you still move quickly when you want to move across the screen.

Drag the **Distance** slider to set the distance (in pixels) the pointer moves before it accelerates. The closer to 20, the longer it takes the pointer to accelerate.

Rearranging your Desktop

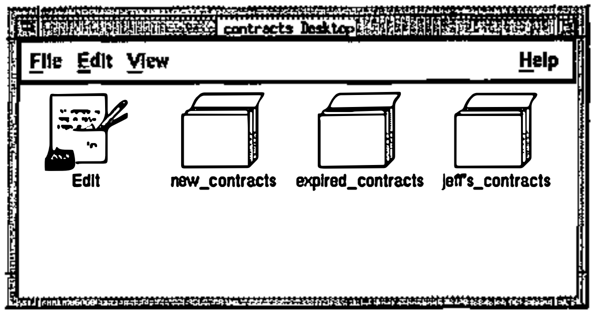




There are several ways you can keep your Desktop and icons organized:

- Arrange Desktop icons in a neat grid pattern by selecting **Clean up** from the Desktop **View** menu.
- Line up icons in neat rows by selecting **Reorganize** from the **View** menu of the window that contains them. Once you reorganize, you cannot return the icons to their original positions except by moving them manually. Because of this, you are asked to confirm that you want to reorganize your icons.
- Move an icon to the Desktop or to another window by dragging it.
- Return icons that have moved onto the Desktop to their original directories by first selecting the icon or multiple icons on the Desktop. Second, select **Put Away** from the Desktop **Edit** menu. The **Put Away** menu is stippled out if no icon is selected. Some icons are *locked onto* the Desktop and cannot be put away. The Accessories, Controls, Applications, and home directory icons are locked onto the default Desktop.

Using more than one Desktop

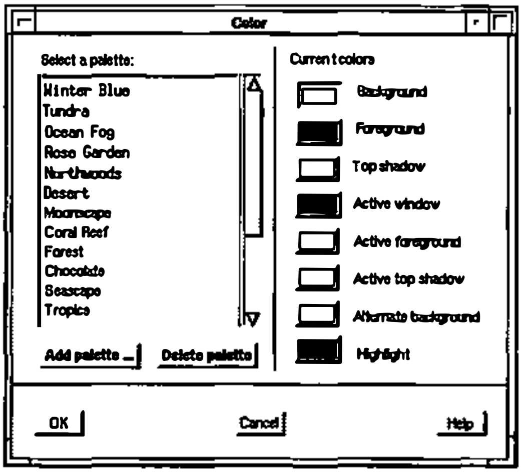


You can set up multiple work environments for yourself by creating different Desktops. This allows you to put only the icons you need for a specific task on the Desktop you use for that task. For example, if you spend part of your time editing contracts and part of your time doing bookkeeping and answering mail, you can set up a separate Desktop for each of these tasks. One Desktop can contain the Edit accessory and directories containing your contract files. The other can contain your bookkeeping programs and records, Mail, and your mail folders. When you open the appropriate Desktop, the materials you need are then at your fingertips.

To create a new Desktop, select **New Desktop** from the Desktop **File** menu. When asked, enter the name for the new Desktop. To create a new Desktop based on an existing one, select **Save Desktop As...** from the **File** menu of the existing Desktop. New Desktops open in separate windows on top of the main Desktop.

NOTE Moving files and directories onto a Desktop is a method of bringing them together in one place so you can work with them more conveniently. However, they only appear to reside on the Desktop. In fact they remain in the directories you got them from. The icon you drag onto a Desktop is not a new copy of the file or directory.

Changing colors



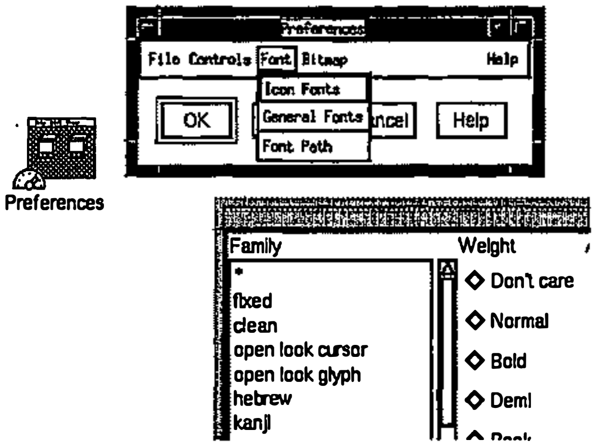
To change the color scheme for all Desktop windows, double-click on the Controls icon to open the Controls window, then double-click on the Color icon to open the window shown above.

Select a palette (color scheme) from the list by clicking on it. If you like the new colors, click on **OK**.

To create your own color scheme, click on **Add palette**. You are asked to name the new palette. The new palette inherits the colors of the current palette. Change the colors in the new palette by clicking on one of the color buttons to the right of the list. In the panel that pops up, select a color from the list, or mix your own color with the sliders. Click on **Apply** when you have the color you want. When you are finished changing colors, click on **OK**. Use the Color **Help** menu to get more detailed instructions.

NOTE Your color change is lost if you click on another color button before clicking on Apply.

Changing fonts



Double-click on the Controls icon to open the Controls window, then double-click on the Preferences icon to open the window shown above.

Select **Icon Fonts** from the Preferences **Font** menu to change the typeface used for icon labels. Select **General Fonts** from the **Font** menu to change the typeface used for window labels and in menus. In either case, a list of fonts like the one shown above is displayed. Select from the list, then click on **Apply** to try the new font.

Use the Preferences control **Help** menu to get more information about changing fonts.

Chapter 7

Working from the command line

In SCO Open Desktop/SCO Open Server, you can control your computer by clicking on or dragging icons and by selecting actions from menus. Such graphical environments (also called *graphical user interfaces* or *GUIs*) were not widely used until computers with the required memory and speed became generally available.

Prior to the development of GUIs, most communication with computers was through the *command line*. A command line is a line on the screen on which you type commands (instructions) to your computer. It is usually identified by a symbol such as " % " or " $ ", called a *prompt*:

```
$ type command here
```

A command line interface is easier for the computer because it only needs to read and react to one line on the screen. It is more difficult for the person using the computer because it requires that obscure commands be remembered and entered in a precisely prescribed way. The command line interface remains a powerful tool because it provides direct access to operating system and networking functionality, and because command line scripts can be used to automate and customize routine tasks.

NOTE The rest of this book is about working from the command line. If you do most of your work on the Desktop, you can skip Chapters 7 through 24 until you need to know how to interact directly with the operating system or networking software beneath the Desktop.

Operating systems

The graphical environment provides a simple, intuitive way for you to work with your computer. Beneath the Desktop, hidden from view, is a complex package of sophisticated software. Your computer does not understand icons and scroll bars; it operates in a language made up entirely of 0's and 1's. Between that binary language and the graphical environment are many layers of software, each more sophisticated than the one it is built upon.

An *operating system* is a group of programs that provide basic functionality on a computer. These programs operate your computer hardware in response to commands like copy, sort, and print. Applications and other programs can use these commands without worrying about the specific signals that make a disk drive work on a particular computer. Because the operating system takes care of such low-level concerns, programs can be more portable and easier to write. An operating system can be seen as a set of functional building blocks upon which other programs depend. It also manages computer resources and resolves resource conflicts, as when two programs want to use a disk drive at the same time.

SCO Open Desktop/SCO Open Server is constructed on top of the UNIX operating system. The UNIX system is used on a wide variety of hardware, ranging from personal computers to supercomputers. It is characterized by its rich assortment of basic tools (or *utilities*), and by its ability to support multiple users running multiple programs at the same time. Programs written to run on UNIX operating systems will run on this software. See Chapters 8 through 13 for information about working with UNIX commands and files.

Another widely used operating system is DOS. DOS was designed to support a single user running one program at a time on a single personal computer. SCO Open Desktop/SCO Open Server can also run most DOS programs. It does this by translating the DOS commands into equivalent UNIX commands. As discussed in the following chapters, you can use either UNIX or DOS commands from the SCO Open Desktop/SCO Open Server command lines, and you can read files from either DOS or UNIX disks. See Chapters 14 through 18 for information about working with DOS commands and files. In addition, see "Using DOS utilities" (page 95) for information about special UNIX commands for working with DOS files.

NOTE Although Chapters 8 through 13 provide a basic introduction to working with the UNIX operating system, Chapters 14 through 18 assume you are already familiar with DOS. The DOS chapters focus on using DOS with this software and the UNIX operating system.

Table 7-1 Similar DOS and UNIX commands

DOS	UNIX system	Action
attrib	chmod	set file attributes (properties, permissions)
cd	cd	change directory
chkdsk	badtrk, fsck	scan disk for errors
cls	clear	clear screen
command	sh, csh, ksh	start a new command processor (shell)
comp	cmp, diff	compare two files
copy	cp, copy	copy a file
date	date	display system date
del	rm	remove a file
dir	ls -l, l	show a long list of filenames
dir/w	lc	show a list of filenames in columns
diskcomp	diskcmp	compare contents of floppy diskettes
diskcopy	diskcp	copy floppy diskettes
edlin	vi, ex, ed, sed	use simple text editor
erase	rm	remove a file
exit	exit	exit current command processor (shell)
fdisk	fdisk	configure hard disk partition
find	grep, fgrep, egrep	search for a sequence of characters in a file
format	format	format a floppy diskette
mkdir	mkdir	make a directory
mode	stty	view or change port settings
more	more, pg	display a file one screen at a time
print	lp	send a file to the lineprinter
ren	mv	rename a file
rmdir	rmdir	remove a directory
sort	sort	sort input file
time	date	display system time
type	cat	display a file
xcopy	cp	copy multiple files and directories
>>	>>	redirect output
<	<	accept input from a file
|	|	pipe output to another command
\	/	pathname separator

Networks

A *network* is a group of interconnected computers. Each computer on the network acts independently, but can transfer information to and from other computers on the network.

A local-area network (LAN) connects computers at one site directly by a high-speed cable, usually an Ethernet™ cable. A wide-area network (WAN), which can be worldwide, connects computers at different sites by transmitting data over telephone lines.

A network might be arranged like this:

This network connects different types of computers running a variety of networking software into a single computing environment. A network like this lets users share the resources of the whole network, which saves time and can eliminate the need to purchase additional hardware and software.

Using the network, you can:

- log in to another computer and use interactive commands such as vi
- execute commands on another computer
- copy files from one computer to another
- exchange mail messages with users on other computers
- share software between computers
- share printers, hard disks, and other devices with other computers

Before you can use any of the networking commands described in Chapters 19 through 24, you must have the required networking software properly installed and configured. Ask your system administrator which networking software your system uses. For networking installation and configuration instructions, see the *Installation and Upgrade Guide* and *Administering Networking Services* in the *System Administrator's Guide*.

Entering commands

After typing a command, press ⟨Enter⟩ to send the command to the computer. For simplicity, typing a command, then pressing ⟨Enter⟩ is also referred to as "entering a command."

Before you press enter, you can use the ⟨Bksp⟩ key (backspace, sometimes labeled with a left-pointing arrow) to back up over and erase previously typed characters. Other command line editing keys may be available, depending on which shell (page 104) you are using and how your system is configured (see your system administrator for details).

UNIX systems do not use the ⟨Del⟩ key to delete text, like DOS computers do. Instead, the ⟨Del⟩ key is used to interrupt programs.

NOTE The DOS and UNIX systems have different conventions for filenames, command options, and wildcards, as discussed in the following chapters.

You can also run most Desktop accessories and applications from the command line (most Desktop controls are specific to the SCO Open Desktop/SCO Open Server environment). The command line names of controls and accessories are given in their glossary entries. See a control's or accessory's manual page for information about using it off the Desktop.

More information about commands

There are hundreds of UNIX commands, and most have many options. Only the most useful commands and options are covered in the following chapters. This book is intended only as an introduction to using the UNIX system. Many books are available on just about any aspect of the UNIX system. For more detailed information about a UNIX command mentioned in this book, see the appropriate manual page. Every UNIX command is thoroughly described in a *manual page* (also called "man page").

To see the manual page for a command, at the UNIX command line type:

```
man commandname
```

Substitute the name of the command for ***commandname*** (you can also use the more graphical **xman** command instead of **man**).

To read manual pages from the Desktop, double-click on *User and Administrator Manual Pages* in the Help Library window. Then double-click on the manual page you want to see in the command summary or table of contents. See Chapter 4 for more about using online help.

(A letter in parentheses following a command or filename refers to the manual page section where the command or file is documented. For example, the **man**(C) command is documented in the Commands section of the manual pages.)

For more about DOS commands, see your DOS manual.

Chapter 8

Getting started

Most of your communication with the UNIX operating system is through the Desktop: you point and click on objects, and the Desktop tells the operating system what to do.

To communicate directly with the operating system, you must enter commands in the UNIX window. Double-click on the UNIX icon to open the UNIX window.

When you finish entering commands, you can close the UNIX window by entering the exit command on the command line. You can also close it like any other window by selecting **Exit** from the **File** menu.

Entering UNIX commands

To enter a command, type its name at the prompt (usually a symbol such as " % " or " $ ") and press ⟨Enter⟩.

Entering a command is like engaging in a dialog with the operating system. You wait until the computer gives you a prompt before entering a command. The new prompt tells you that the operating system has finished processing your previous command and is ready for another.

Most UNIX commands have options to modify their behavior. Many commands also take arguments, on which they act. The command comes first, then the options (which are usually indicated by a " - " in front of them), then the arguments, as in the following example:

```
sort -r myfile
```

In this example, sort is the command, **-r** the option, and **myfile** the argument.

The option **-r** specifies that the sort is done in the reverse of normal order; " - " indicates that the letter " r " is an *option,* which sets some sort of special condition for the program. (Different programs have different options.)

The argument **myfile** is the name of the file, the contents of which are to be sorted.

Note the UNIX system is *case sensitive*: that is, it assumes that **SORT**, **Sort**, and **sort** are different commands. Most UNIX commands are all lowercase.

Specifying command input and output

Many UNIX commands require *input* and *output;* that is, some information to read and process, and somewhere to store the results. If you do not tell a command where to find its input and output, it makes assumptions about where to read and write information; it uses the *standard input* and *standard output.* (These are, respectively, your keyboard and your screen, which is why information is read from and written to your terminal unless you tell a command to use another destination.)

In the example, **sort -r myfile** gets its input by opening the file named *myfile.* Because no output destination is specified, **sort** sends its results to the standard output destination, which is normally your screen.

You can redirect commands' input and output by using the symbols " < " and " > " on the command line, followed by the name of the file to read or write. For example:

```
sort < file1 > file2
```

makes sort treat *file1* as its input, and send its output to *file2* (that is, the contents of *file1* are sorted and the results are placed in *file2*.)

If you send the output of a program to a file that already exists, the existing file is deleted and replaced by a new file with the same name, containing the output of your program; that is, the contents of the existing file will be overwritten.

To add the output of a command onto the end of a file (known as *appending* the output), type ">>" instead of " > ". For example:

```
sort < file1 >> file2
```

appends the output from **sort** onto the end of *file2*.

Running a sequence of commands

There are three ways to run a sequence of commands:

- You can run them individually by typing them at the prompt, either on separate lines or on the same line separated by semi-colons; for example:

  ```
  ls
  pwd
  who am i
    or
  ls; pwd; who am i
  ```

- You can use an editor to store them in a script file, then run the script (see "Running command sequences" (page 99) for further details).
- If the commands all operate on the same data file one after another, you can run them as a *pipeline.*

A pipeline is a sequence of commands that run, one after another, on the same data. The output of the first command is sent to the second command via a *pipe*. The pipe is represented by the symbol " | ". For example:

```
sort filename | uniq
```

uniq is a program that reads lines from its input, copies them to its output, and eliminates duplicates; that is, if it reads the same line twice it ignores that line the second time. This pipeline sorts the lines in *filename*, then sends the output from **sort** through a pipe to **uniq**, which then sends its own output to the standard destination (in this case, your screen).

You can stack commands up in a pipe by separating them with a " | " symbol, and you can send the final output of a pipe into a file with the ">" symbol. For example:

```
sort filename | uniq | wc > words
```

This pipeline creates a file called *words*, containing a count of all the words that occur on non-identical lines in *filename*. (**wc** is a program that counts the number of words, lines and characters in its input.)

Aborting a command

Press the ⟨Del⟩ key.

This is a quick way of recovering from a command that is still being executed.

Pressing the ⟨Del⟩ key sends the **INTERRUPT** signal. (Some systems use the ⟨Ctrl⟩C key for this purpose.)

Some programs deliberately ignore the ⟨Del⟩ key, such as the **vi** editor, the shells which process your commands, and the UNIX window (**scoterm**). These programs understand the ⟨Del⟩ keystroke as an instruction. For information about shells, see "Changing your shell" (page 104).

Chapter 9

Finding your way around

Files are stored in *directories*. A directory can contain files or other directories. Collectively, all directories and the files they contain are called a filesystem. The UNIX operating system provides a variety of tools for navigating the filesystem from the command line.

Identifying the current directory

To identify your current directory, type **pwd** and press ⟨Enter⟩.

The current directory is the one you are currently working in. **pwd** stands for *print working directory*. The word *print* is used instead of *display* because the UNIX system was developed in the days of teletype terminals, when all output was printed.

Viewing the contents of a directory

To view the current directory's contents, type **ls** (short for *list* files) and press ⟨Enter⟩.

ls lists the files and directories in the current directory.

To view the contents of a different directory, enter:

ls *directory*

where ***directory*** is the name of the directory whose contents are to be listed. See "Specifying directory names" (page 73) for more about referring to a directory other than the current one.

To modify the format or kind of information displayed, the following related commands can be used:

l	give a detailed listing of all filenames and their attributes
lc	give a listing of all files, split into columns
lf	give a listing of all files, split into columns, with directories followed by a slash (/) and runnable programs followed by an asterisk (*)
lr	list the contents of the current directory. If it contains any subdirectories, list their contents afterwards. If they contain subdirectories, continues listing their contents indefinitely. (The " r " is for *recursive*; for more about directories, see "Specifying directory names" (page 73).)
lx	give a listing of all files, ordered in rows

See the ls manual page for more about these commands.

Viewing the contents of large directories

The contents of directories that contain many files will scroll past faster than you can read them. To view the list one page at a time (where a "page" means as much text as you can put in the UNIX window at any time), enter:

```
ls | more
```

After each page **more** will display `more`; to see the next page, press ⟨Space⟩.

In this command, ls is piped (|) to the **more** command. See "Running a sequence of commands" (page 69) for more information on using pipes.

Specifying directory names

The UNIX system stores files in *directories.* A directory can contain files or other directories. Each user has a *home* directory. You can keep your personal files in your home directory, or create subdirectories within it to store files relating to certain categories. For example, your personal directory structure might look like this:

`/u/charless`	the home directory; contains ...
`/u/charless/.profile`	system file
`/u/charless/.login`	system file
`/u/charless/mailfolders`	subdirectory containing mail folders
`/u/charless/mailfolders/stuff`	a mail folder
`/u/charless/mailfolders/mbox`	another mail folder
`/u/charless/work`	subdirectory containing work files
`/u/charless/work/proj1`	a file called *proj1*
`/u/charless/work/proj2`	a file called *proj2*

/u/charless is the home directory. It is located in the */u filesystem.* Within */u/charless* there are some system files (*.profile* and *.login*) and some directories (*mailfolders* and *work*). Each of the directories contains files partitioned according to their purpose; this makes it easier to keep track of a large collection of files.

NOTE This example is simplified and is intended to show the structure of a home directory, not an actual listing provided by a program such as l.

The filesystem structure resembles an upside-down tree; each branch is a directory or subdirectory, and each leaf is a file. The main stem is known as the *root* directory. Each directory and file can be identified by a unique path in the tree.

To specify the path to a directory or file that is not in the current directory, you must give either an *absolute* or a *relative* path.

An absolute path lists all the directories and subdirectories you must enter to reach the target file, starting at the root (/) directory. For example:

/usr/fred/work/target

refers to the *usr* directory within the *root* directory, which contains the subdirectory *fred,* which contains another subdirectory, *work,* which contains the file or directory called *target* for which you are looking.

Build a relative path by specifying all the directories and subdirectories you must enter to reach the target file, starting from your current position in the filesystem. (To find out where you are, see "Identifying the current directory" (page 71).)

For example, supposing you are in */usr/me,* and want to specify a path to */usr/fred/work/target*. You can get there by the relative path:

../fred/work/target

The " .. " symbol in your current directory represents the *parent* of the current directory, or the directory which is one level closer to the root directory; for example, *fred* is the parent of *work.*

You may also see a "." symbol in a listing of your current directory. This refers to the current directory itself. For example, to refer explicitly to a file called *filename* in your current directory, you could type *./filename* or *filename.*

Finding a file

To check if a file is in the current directory, type **ls myfile** and press ⟨Enter⟩. If it is present, the file name is displayed.

You can search for more than one file in a directory, but the files should have similar names for this to be practical. For example, supposing the current directory contains *my, myfile1, myfile2, myfile3, myfile4* and *myfile10.* To find all files that have names starting with *my,* type **ls my*** and press ⟨Enter⟩.

The names of all files in the current directory that begin with *my* are listed. When you append an asterisk (*) wildcard character to the partially defined filename in the command, the system expands this to match filenames in this directory that start with *my* and are followed by zero or more characters. The asterisk wildcard "matches" any sequence of characters, including none at all, so *my** matches *my* as well as *myfile.*

To locate files with only one additional variable character, use the " ? " wildcard character. For example, type **ls myfile?** and press ⟨Enter⟩.

The system displays *myfile1, myfile2, myfile3* and *myfile4,* but not *myfile10.*

Typing **ls myfile??** and pressing ⟨Enter⟩ results in the filename *myfile10* being displayed.

To locate files with a range of characters in their names, enter (for example):

```
ls myfile[1-5][1-5]
```

This will list all the files starting with *myfile,* followed by two digits in the range one to five. (You can also specify a range of letters, for example [A-Z] or [a-z], and you can create sets of characters for which to search. For example [A-C1-90] matches the capital letters A, B, C or any digit.)

Locating files outside your current directory

To locate a file that is *not* in your current directory, for example, *myfile*, enter:

```
find / -name myfile -print
```

find searches the specified directory and all directories branching from it. The direction of the search can be from the root (/) of the filesystem structure, as in the example above, or relative to your current position in the filesystem (or wherever you specify).

-name and **-print** are options to the **find** command for specifying the name of the file to search for and printing (displaying) the names of all matching files on the screen.

Changing directories

To change directories to a subdirectory of the current directory, enter:

```
cd mydir
```

In the example above, *mydir* is called a relative pathname because directions to it are given relative to the current directory. You may also use absolute pathnames with cd. Absolute pathnames specify directions to a directory from the top-level (root) directory. See "Specifying directory names" (page 73) for instructions on using pathnames.

To return to the previous directory from *mydir* using the relative pathname method, enter:

```
cd ..
```

This takes you back to the parent directory.

Creating directories

To create a directory called *newdir*, enter:

```
mkdir newdir
```

You may use either absolute or relative pathnames. The example above creates a subdirectory called *newdir* in the current directory.

Directories are created with access permissions that can be modified by the owner. For more information on permissions, see "Changing access permissions" (page 80).

To create a directory, you need to have write permissions for the directory within which you wish to create it, and your new directory name must be unique in this directory.

Removing directories

To remove a directory called *olddir*, enter:

rmdir olddir

You can not remove a directory with **rmdir** unless it is empty (contains no files or subdirectories) and you have write permission. See "Removing files" (page 88) and "Changing access permissions" (page 80).

You can not remove a directory if you are in it. To remove the current directory, first change to another directory.

Chapter 10
Working with files

A file is the basic unit in which the UNIX system stores information. While you may see references to sub-units, such as records and fields, the file is the smallest block of information that is stored by name and recognized by the UNIX system.

Although most of your work with files is done with Desktop tools (such as Edit), the UNIX system provides a rich assortment of tools for manipulating files from the command line. Tools are available to work with files (moving, copying, deleting and changing them) and to work within files (searching for information, editing, and sorting).

Moving and copying files

When you *move* a file, you are placing it in another directory (or under another name in the same directory). When you *copy* a file, you are creating a duplicate, which occupies additional space in the filesystem. When you create a new *link* to a file, you are giving the file an additional name: the file is stored in only one space, but the linked names may appear in several directories.

Creating files

You create a file whenever you make a copy of a file, edit a new file, or cause the output of a command to be directed to a file that does not yet exist. See "Specifying command input and output" (page 68).

Although you will usually use Desktop editors to create or change files, you can edit files from the UNIX command line with the **vi** editor. See "Using vi" (page 105).

Moving files

To move a file to another directory, enter:

mv *filename destination*

where ***filename*** is the name of the file you want to move (preceded by its path if it is in a directory other than the current one), and ***destination*** is the path to the directory where you want to put it. The file disappears from it's original directory and reappears in the destination directory.

To change the name of a file within the current directory, enter:

mv *old new*

where ***old*** is the file's current name, and ***new*** is the file's new name.

You can combine these techniques to move a file to a different directory ***and*** give it a new name at the same time. For example:

mv chapter.1 /u/workgroup/finished.chapter.1

moves *chapter.1* to */u/workgroup* and renames it *finished.chapter.1* at the same time. You can only use this technique to move files if you have write permission to them.

Copying files

To make a copy of a file, enter:

cp *old new*

where ***old*** is the name (preceded by its path if it is in a directory other than the current one) of the file you want to copy, and ***new*** is the name for that copy.

new does not have to be in the same directory as ***old***. See "Moving files" (this page).

Combining files

You can combine two files, end to end, using the **cat** command. **cat** simply copies its input to its output. You use it like this:

cat file1 file2 > file3

where *file1* and *file2* are files to read (the input) and *file3* is the name of the file to create (cat's output). **cat** reads *file1* writing a copy of it to *file3*, then reads *file2*, appending it to *file3*.

Linking files

To create a *link* to a file, enter:

ln file1 file2

A *link* is a new file name (or "link") that refers to a file that already exists; in effect, the file has two (or more) names. The names, or *links*, do not need to be in the same directory, or have the same owner.

You can use links as a shortcut to edit a file in another directory, by creating a link to the file and keeping the link in your home directory. Then, whenever you want to edit the file, instead of changing to the other directory, you can just edit the link.

You can also use links as a shortcut when changing directories. Create a link to a directory using the **-s** (symbolic) option, and **cd** into the link. For example, suppose you work in */u/workgroup/tasks/project/01* and your home directory is */u/me*. Your normal command to work on a file is

cd /u/workgroup/tasks/project/01

but you can create a link:

ln -s /u/workgroup/tasks/project/01 01

that creates a link in your current directory. Then you can move around like this:

```
$ pwd
/u/me
$ cd 01
$ pwd
/u/workgroup/tasks/project/01
```

The **-s** option means that the link is *symbolic*; it points to a file on a different filesystem, or to a different type of file (such as a directory).

See "Removing files" (page 88) for instructions for removing a link.

Changing files

You can change files by altering their names, changing their attributes (permissions or properties), and working on their contents (for example, by sorting them). These operations can be performed on the Desktop, but can also be accomplished from the UNIX command line.

Renaming files

To rename a file, use the **mv** command. For example, the command:

```
mv file1 file2
```

renames *file1* to *file2*.

You can use this technique to rename directories if you have write permission to them.

Changing access permissions

If you cannot look at a file, you probably do not have *read* permission for that file.

To find out if you have read permission, enter:

l ***filename***

If the second character position in the ten character field at the left of the listing is "r", the owner of the file has read permission on the file. The login name of the file's owner is listed in the third field in the listing. If you are the file's owner but the left (owner) read permission is not set, you can give yourself read permission like this:

chmod u+r ***filename***

This **chmod** command modifies the permissions on ***filename*** so that the owner (denoted by "u" for user) is given read permission (denoted by the "r").

Only the owner of a file can use **chmod** to alter the permissions on that file.

To give yourself permission to write to or *execute* (run) a file, use either the **chmod u+w** or the **chmod u+x** version of the command.

You can give members of your work group permission to read, write and/or execute the file using the **g+r**, **g+w**, or **g+x** versions of the command, if you own the file.

You can make a file publicly accessible using the **o+** form of the command ("o" is for *others*, meaning all other users of the system.)

To revoke permission to read, write or execute a file for a type of user, use a "-" instead of a "+"; for example, to revoke read permission for other users, use:

chmod o-r ***filename***

Sorting files

To sort a file containing lines of text or numerical data in a variety of ways, use **sort**. For example, to sort a file (called *filename*) containing lines of text into dictionary order, regardless of upper- or lowercase:

```
sort -df < filename > sorted
```

The **-d** option specifies that "dictionary" order is to be used for the sorting process. The **-f** option means that lowercase letters are folded into uppercase (capital letters) for purposes of comparison. The file called *sorted* contains the result of the sort operation on *filename.*

To combine two files into one new file, the contents of which are sorted, enter:

```
sort -u file1 file2 > file3
```

This creates a file called *file3,* containing the sorted, merged contents of *file1* and *file2.* (**sort** sorts the files as it merges them.) The **-u** option tells **sort** to make sure that each line in *file3* is unique; that is, if both *file1* and *file2* contain an identical line, only one copy of the line will be written to *file3.*

The **-r** option reverses the order of the sort. Use the **-n** option when sorting lists of numbers, so non-numeric characters in the numbers (minus signs, decimal points, and leading spaces, for example) are not sorted incorrectly. The **-M** option makes **sort** assume that the first three characters of the field being sorted are months (like JAN, FEB, MAR, and so on) and sort them into date order.

You can make **sort** pick any portion of a line on which to base its comparisons. For example, to sort a list of names followed by months on the basis of the month:

```
sort -M +1 < filename > sorted
```

will sort a file containing lines like

```
martin FEB
angela DEC
judith JAN
```

into

```
judith JAN
martin FEB
angela DEC
```

The **+1** option tells **sort** to make comparisons between lines on the basis of the second field of each line. (A field is a sequence of characters separated by spaces or tabs; **sort** counts fields starting from zero.) So the "month" abbreviation on each line of the file is used as the basis for the sort operation above.

If you have a file where data records are made up of fields separated by some special character like a colon " : " (called a *separator*), you can tell **sort** to use a different separator by using the ***-tseparator*** command.

For example, **-t:** causes **sort** to split lines into fields separated by colons.

Looking inside files

The UNIX system does not normally distinguish between types of file. There are many different types of files in the filesystem, some of which you can work on and some of which you should avoid. For example, you can edit text files with the **vi** editor, and you can also read in and edit program files with it, although this is not a useful thing to do. It is more efficient to use the specific UNIX tools for identifying the type of information files contain.

Identifying file type

You can find out what type of information a file contains using:

```
file filename
```

file looks at the contents of a file and tries to determine what type of information the file contains. **file** can tell whether it is an executable program, contains data used by a program, or is text in English or another language.

It is a good idea to use **file** before examining the contents of a file as described below; if you try to examine a binary (or executable) file you may render your display unreadable, because binary files often contain characters that are interpreted as control codes by the terminal.

Previewing files

To look at the first or last lines of a text file, use **head** or **tail**. For example:

```
head filename
```

displays the first ten lines of ***filename***, while:

```
tail filename
```

displays the last ten lines of ***filename***.

If you use a numerical option, for example **-20**, **head** or **tail** will display 20 lines instead of 10.

Viewing very short files

You can look at the contents of a short file, using:

cat ***filename***

cat con*cat*enates its input to its output. This means that **cat** sends ***filename*** to your window, for you to read. (As your output is usually your window, **cat** sends its input there unless you tell it to use an output file).

You can **cat** more than one file at a time; the files are listed one after the other.

Hint: if you do not know what is in a file you want to **cat**, try using:

cat -v ***filename***

This will cause any unprintable characters in ***filename*** to be displayed in a manner that will not corrupt your window.

See "Combining files" (page 78) for more information about **cat**, input, and output.

Viewing longer files

To look at the contents of a file that is too big to fit in a single window, enter:

more ***filename***

more displays files one page (window) at a time. After filling the window with text, **more** will display the prompt `more`. To see more, press ⟨Space⟩.

While **more** is running, you can search for text in a file by entering a " / " followed by some text to find; for example:

/something

will make **more** search forward until it comes to the next occurrence of the word "something".

If you accidentally read past a piece of text you want to look at, press b to jump back a page.

To quit **more**, type **q**.

There are a variety of other commands which **more** recognizes. If you want a list of them, press **h** for help when you are viewing a file using **more**.

Searching files

If you want to find which file contains some specific word or words, there are UNIX tools to help you. You can search files rapidly to locate a known piece of text, or you can search files more slowly to locate a piece of text when you are unsure of the spelling.

Searching for text

To search a file for a specific piece of text, enter:

> **fgrep** ***text file1*** **[*file2 file3* ...]**

fgrep ("fast grep;" see "Using wildcards" (this page) for more about **grep**) searches all the files in ***file1***, ***file2*** and so on for the specified ***text***, and reports any matches.

To search for a text containing spaces or tab characters, enclose the text in quote marks.

To search for a text containing quote marks, put a backslash immediately in front of each quote character within the text.

If you are not sure whether the text is uppercase, capitalized, or all lowercase letters, type **fgrep -y**. **fgrep** will then ignore the case, and report all matches.

To see all lines in a file that do *not* contain the text, type **fgrep -v**.

Using wildcards

You can search files for text when you are unsure of the spelling of the text by using wildcards and **grep**. (**grep** is an acronym for "global regular expression print;" a "regular expression" is a complex wildcard.) For example, to search a file for the word "center" when you are not sure whether it is spelled the American way ("center") or the British way ("centre"), or even if it is present in another form ("central" or "centrally"):

```
grep 'cent[er]' filename
```

The "[er]" is a *set* containing the characters " e " and " r ". This means that **grep** will search for the text "cent" followed by either an " e " or an " r ". (The single quotes are needed because otherwise "[er]" will be interpreted as a wildcard by the shell, before the command is passed to **grep**.)

If you know that you want either "centre" or "center" but not "centrally" or some other combination, you can search for:

```
grep 'cent[er][er]\ ' filename
```

The "\ " means "followed by a literal space character." (Remember, you need to enclose the search text in single quotes because it contains a space.) So 'cent[er][er]\ ' will match "centre " or "center " but not "centerpoint " because the two characters are not followed immediately by a space.

You can put more than two characters in a set. For example, you can search for any character in the set [ABCDEabcde]. To save typing, you can enter this set as [A-Ea-e], where the " - " indicates that your wildcard is a *range* of characters. Alternatively, instead of using the "[...]" set notation, you can search for the special set " . " (that matches any character except a newline).

To search for a sequence of characters in a set that repeats an indefinite number of times, you can use the " * " symbol after the set of characters for which you are searching. " * " matches zero or more occurrences of the preceding wildcard. For example:

```
grep 'cent.*' filename
```

searches *filename* for the letters "cent" followed by a sequence of zero or more characters matched by " . " (the set matching any single character). Because " . " matches everything except a newline, this wildcard will match "center", "center", "central", "accentuate" and "centipede".

You can also match anything that is *not* part of a set. If you want to find every line that contains a "cent" unless it is in "centipede" use the following command:

```
grep 'cent[^i]' filename
```

The "^" at the beginning of the set means "match any character *except* a member of this set."

You can search for text at the beginning or end of a line, using the "^" or "$" symbols respectively. For example:

```
grep '^begin' filename
```

searches for all lines beginning with the string "begin" while:

```
grep 'end$' filename
```

matches all lines with the string "end" as a terminator.

Finding out how large a file is

You can find out how large a file is by using the wc command:

```
$ wc file
    684    4380    32675   file
```

wc counts the number of lines, words, and characters in a file (in that order). You can use wc to get specific totals in any order using the options **-l**, **-w**, or **-c** to stand for lines, words or characters respectively. For example, to see the number of characters and lines in a file (in that order):

```
$ wc -cl file
    32675  684  file
```

You can also give wc a list of files to count. For example:

```
$ wc chap1 chap3
    105    676   3844 chap1
    675   3869  24269 chap3
    780   4545  28113 total
```

Extracting fields

If you have a file containing columns of data in textual form, you can extract information from it using a variety of tools. For example, suppose you have a file called *blackbook* containing names, extension phone numbers, login names and dates, in a format like this:

```
Michael Stand:571:mikes:JAN-1-91
Sue Penny:284:suep:FEB-6-89
Joshua Ford:954:joshp:JUL-30-88
Liz Addams:553:lizh:AUG-15-91
                ...
```

To see Sue Penny's record, use the following command:

```
$ grep Sue blackbook
Sue Penny:284:suep:FEB-6-89
```

This is hard to read. To see only Sue's extension number (the second field), you can use the **cut** command:

```
$ grep Sue blackbook | cut -f2 -d:
284
```

cut extracts individual fields from a file containing records on separate lines. The **-f2** option tells **cut** to extract only the second field of each record; the **-d:** option means that fields are delimited with a colon (:) instead of tabs.

The " | " is called a *pipe;* it tells **grep** to send its output to another program (in this case, to **cut**) instead of the window (or "standard output"). See "Combining files" (page 78) for information about pipes, input and output.

To see a list of all the people in your file, followed by their logins, you do not need to use **grep**. Just use **cut**:

cut -f1,3 -d: blackbook

The **-f1,3** option tells **cut** to extract the first and third fields in each record:

```
Michael Stand:mikes
Sue Penny:suep
Joshua Ford:joshp
Liz Addams:lizh
```

To alphabetize your list, you can **sort** it like this:

```
$ cut -f1,3 -d: blackbook | sort -df
Joshua Ford:joshp
Liz Addams:lizh
Michael Stand:mikes
Sue Penny:suep
```

Printing a file

To print a file, enter:

lp filename

lp responds with

```
request id is laserwriter-635 (1 file)
```

This command sends *filename* to the print queue. (**lp** is short for "line printer".) The "request id" line means that the file will be printed on the printer named "laserwriter", and has the job number laserwriter-635.

A print queue is a queue of files waiting to be printed on a specific printer. Because a UNIX system may have many users, any or all of whom may be printing files, your file might not be printed at once. It goes onto the back of the queue. However, if it is the only file waiting to be printed, it will be processed at once. Otherwise, it has to wait for its turn.

If you know that several printers are connected to your system, and you want to send a file to a printer that is not busy, you need to know the destination printer's name. Use the **-d** option to **lp** to choose the destination; for example:

lp -d fast_printer filename

sends *filename* to the printer named *fast_printer*.

You can get a list of the printers available to you by using the **lpstat** (line printer status) command:

lpstat -s

To cancel a print request, if it has not yet printed, use the **cancel** command and the request-id you were given when you entered the **lp** command to print the file. For example:

cancel laserwriter-635

cancels the print job "laserwriter-635". If you do not remember the request-id, enter the **lpstat** command with no options to see a list of print jobs in the queue.

Removing files

It is necessary to remove files from time to time, to prevent the filesystem from filling up. However, you should be extremely careful about removing files. Although the Desktop environment lets you retrieve deleted files, once a file has been removed it is gone forever; there is no way to get back any information that you have lost. Therefore, you should use the **rm** (remove) command with care.

Removing ordinary files

To delete a file, use the **rm** command. For example:

```
$ rm -i filename
filename: ?
```

rm -i is *interactive;* when you see the question mark, you can either type " y ", in which case it will destroy *filename,* or " n ", in which case it will not.

It is a good idea to use the **-i** (interactive) option with **rm** because once you have removed a file, it is impossible to get it back again.

To remove several files, you can type **rm -i** and use wildcards to select them. (Always use the **-i** option with **rm *** unless you are absolutely certain that it is safe to delete everything in the current directory.)

Removing links

You can remove a link you no longer want, just like a normal file, with **rm.**

rm link

deletes a link called *link,* just like a normal file.

The file itself will not be deleted until the last link (or name) by which it is referred to is deleted.

Searching for lost files

If you have put a file somewhere and cannot remember where it is, you can use the **find** command to locate it:

> **find / -name** ***filename*** **-print**

The " / " tells **find** to start searching in the *root* directory. **find** searches its starting directory, and all the subdirectories it can find, in order. If you know your file is in one of your own subdirectories you could tell **find** to start searching from **$HOME** instead of " / ". **$HOME** represents your home directory; see "Setting variables" (page 101) for details.

The **-name** option is followed by the name of the file for which you are looking. Every time **find** sees a file with this name, it will carry out the actions specified by the subsequent options.

The **-print** option tells **find** that the action to take when it finds **-name** ***filename*** is to print (display) its full path and name on your screen.

find can carry out other tasks besides showing a file's full path. For example, the command:

> **find /bin -name filename -exec l {} \;**

causes **find** to execute l on any file it finds with the name *filename* under the directory */bin*. (The " {} " in the **-exec** command stands for the name of the found file; the " \; " marks the end of the exec option.)

> **find / -name chap3 -print**

If the command above results in a series of error messages like:

```
/u/charless/stuff/odt/os/chap3
/u/w/Xenix/OS2.3.2/Intlsupp/Guide/chap3
find: cannot chdir to /etc/conf/pack.d/arp
find: cannot chdir to /etc/conf/pack.d/arpproc
find: cannot chdir to /etc/conf/pack.d/atb0
        ...
```

you can ignore the errors by re-issuing the command as follows:

> **find / -name chap3 -print 2> /dev/null**

The error messages are coming from **find**'s *standard error*, because **find** does not have permission to read these directories on your behalf. The fragment

> **2> /dev/null**

tells **find** to send its error output (the *standard error*) to */dev/null*. (*/dev/null* is a *device file* that suppresses any output stream you send to it.)

In general, if any program gives a series of error messages, you can stop them from cluttering up your window by adding "2> /dev/null" to the end of the command line, or you can add them to a log file by adding "2>> error.file" to your command. (Note that this will not work if you are using the C shell rather than the Bourne or Korn shells.)

Chapter 11
Copying files to disk and tape

Most of the time, you work with files in the filesystem, which is stored on your computer's hard disk. However, sometimes you may want to copy files to and from tapes or floppy disks; for example, you may want to give a copy to a user on a machine not connected to your own, or to store infrequently-used material on a tape (which is much cheaper than space on the filesystem), or to make a back-up copy of your work.

There are many ways of saving and retrieving files from floppy disk or tape. The simplest method is to use a **tar** archive, as discussed in "Creating a tar archive" (page 92). Other methods are more complex or are only available to the system administrator.

You can also copy files to and from DOS floppy disks relatively easily. To use a DOS floppy disk, you must format the disk for DOS (page 95) if it is not already formatted, then use the DOS commands described in "Using DOS utilities" (page 95) to copy files between the UNIX filesystem and DOS floppy disks.

Formatting floppy disks

You must format a floppy disk before you can use it. To format a disk, ensure that it is in the appropriate drive and then enter:

format
or
format ***drive***

If you do not specify the device, the default drive will be used.

drive is the *device file* the UNIX system uses to communicate with the type of disk you are creating. (The UNIX system sees all pieces of equipment attached to the computer as files; it communicates with them by reading from and writing to a special *device file* stored in */dev*.) There is a different device file for each different format of floppy disk.

You determine the name of the device file to use as follows:

1. All floppy disk devices are located in */dev* and begin with *rfd* (the " r " is short for *raw*, because the UNIX system has to access the disk directly).
2. If your computer has only one floppy disk drive, follow this with a number " 0 ". If your computer has two or more drives, you can follow it with a " 0 ", " 1 ", or higher number (depending on whether you want to format a disk in the first, second or subsequent drive).
3. Follow this digit with the number of tracks per inch on the disk; for example, 135 if it is a high-density 3.5-inch floppy. (The number of tracks per inch, or "tpi", should be indicated on the disk or its case.)
4. Follow this number with either "ds" if the disk is double-sided, or "ss" if it is single-sided.
5. Finally, finish the device name by adding the number of sectors per track; 9 if it is a low-density 5.25-inch or 3.5-inch floppy, 15 if it is a high-density 5.25-inch floppy, or 18 if it is a high-density 3.5-inch floppy.

For example, to format a disk in the second floppy disk drive that is to be a high-density 3.5-inch double-sided disk, enter **format /dev/rfd1135ds18.**

format will prompt you to insert the floppy and press ⟨Enter⟩ to continue.

Note that it takes time to format a floppy disk — typically a minute or so (although this may vary).

Creating a tar archive

A **tar** archive can be thought of as a special file that contains other files and their associated directory information. To create a **tar** (tape archive) file on a floppy disk, enter:

```
tar cvf device filename
```

where ***device*** is the device file corresponding to the floppy disk, and ***filename*** is the name of the archive. (See "Formatting floppy disks" (page 91) for information about device files.)

tar lets you use an abbreviation to specify the device to use. To see the list of available types, enter:

```
$ tar
Usage: tar -{txruc}[0-9vfbkelmnpwAF] [tapefile] [blocksize] [tapesize] files...
        Key     Device           Block    Size(K)    Tape
        0       /dev/rfd048ds9   18       360        No
        1       /dev/rfd148ds9   18       360        No
        2       /dev/rfd096ds15  10       1200       No
        3       /dev/rfd196ds15  10       1200       No
        4       /dev/rfd096ds9   18       720        No
        5       /dev/rfd196ds9   18       720        No
        6       /dev/rfd0135ds18 18       1440       No
        7       /dev/rfd1135ds18 18       1440       No
        8       /dev/rct0        20       0          Yes
        9       /dev/rctmini     20       0          Yes
```

Using this list, you can select the size of disk you want to use. For example, to create a tar file on a 720K floppy disk in the second floppy disk drive, just use the command **tar cv5** ***filelist*** (where ***filelist*** is a list of files to create in a tar archive on */dev/rfd196ds9,* separated by spaces).

If you do not specify a device, the **tar** archive will be created on the default device (specified in */etc/default/tar*).

If you specify a ***filename*** instead of a device name with the **-f** option, **tar** will create the archive in your current directory (or the path indicated by ***filename;*** it looks like a special type of file, called a *tarfile.* The **tar** archive will be created and copies of all the files will be stored in it.

Listing tar archives

Type **tar tvf** ***device*** and press ⟨Enter⟩. (***device*** is either the name of the device containing the floppy or tape where the archive is stored, or the name of the file containing the archive.)

Extracting tar archives

To extract files from a **tar** archive, type tar **xvf** ***device filelist*** and press ⟨Enter⟩. tar looks inside the archive on ***device*** (or the file of that name) and extracts any file it sees which matches ***filelist.***

For information about other options to tar, see the **tar** manual page.

Understanding magnetic tape

You can copy files to and from tape devices using **tar**, in the same way as you deal with floppy disks. However, there are a number of differences between tape and floppy disk systems. Notably, although magnetic tapes can store far more data than a floppy disk, they can only provide *serial access* to the information. That is, when reading or writing a tape you must start at the beginning and read through each file stored on it in order until you get to the end — you generally cannot jump around or skip files. Consequently tapes cannot be used as filesystems.

To copy files to and from a tape device you should use **tar**, with the appropriate device file (from the list below). You may also need to use the **tape** command to control the tape drive directly; see "Rewinding and erasing tapes" (this page).

There are several different types of tape which may be available. The most common are:

QIC-02	A full-sized quarter-inch tape cartridge, the first QIC-02 drive uses the */dev/rct0* device file.
QIC-40/QIC-80	These smaller mini-cartridge units related to the QIC-02 format are accessed through the */dev/ft0* device file.
mini-cartridge	Mini-cartridge tape drives are linked to the floppy disk drive controller and differ significantly from the QIC family of tape drives. Notably, you must format mini-cartridge tapes before using them. They are accessed via the */dev/rctmini* device.
SCSI	SCSI tape drives are controlled by a SCSI controller, like SCSI hard disks. They are accessed via the devices named */dev/Stp0* to */dev/Stp3* (or */dev/rStp0* to */dev/rStp3* for raw access).

For further information see the chapter on tape drives and controllers in the *Hardware Configuration Guide.*

Rewinding and erasing tapes

To rewind or erase a tape, you should use the **tape** command.

To rewind a tape, enter:

```
tape rewind
```

It is a good idea to rewind the tape to the beginning after every use, or after encountering an error.

To erase a tape, enter:

```
tape erase
```

It is not necessary to erase a tape before reusing it. However, you may want to erase a tape for security reasons.

You should retension any tapes that you use regularly, or that have been in storage and from which you now wish to read. This takes up any slack in the cartridge and reduces the likelihood of errors. The command to retension a cartridge is **tape reten**.

In addition, you should write-protect your tapes to prevent accidental erasure or overwriting. This is done by turning the slot on the cartridge to the SAFE position; turn it the other way when you intend to write over or erase the tape.

Formatting a DOS floppy disk

The UNIX system provides special tools for manipulating floppy disks that are compatible with DOS.

To format a floppy disk for use with DOS, enter:

dosformat ***device***

where ***device*** is a special file, as explained in "Formatting floppy disks" (page 91).

A DOS format disk cannot be used with **tar**; to store files on it you must use the special DOS utilities described in "Using DOS utilities."

Using DOS utilities

Several special UNIX commands are provided for manipulating DOS disks (not to be confused with the actual DOS commands provided by SCO Open Desktop/SCO Open Server DOS Services). They are as follows:

dosls ***drive***
: provides an **ls** style listing of the files stored on ***drive***, where ***drive*** is either a UNIX-style device file or the DOS drive name (A: or B:)

dosdir ***drive***
: similar to **dosls**, but provides a directory listing after the style of the DOS program **dir**

doscp ***filename destination***

copies the file ***filename*** to the specified ***destination***. You can copy files to a subdirectory on the DOS disk, if you specify the pathname (for example, *A:\MYDIR\MYFILE.TXT*). You can use **doscp** to copy files to a DOS disk from a UNIX system, or to a UNIX system from a DOS disk.

Note that **doscp** does not recognize wildcards; if you want to copy more than one file using wildcards, you should enter the following:

```
for file in wildcard
do
doscp $file destination
done
```

where ***wildcard*** is used to identify the files you want to copy, and ***destination*** is where you want to copy them.

dosrm ***filename***

deletes the named file. Note that you can give a pathname, if the file is in a subdirectory on the DOS disk.

DOS filenames are different from UNIX filenames. The following rules apply:

case	All DOS filenames are uppercase. UNIX files are converted to uppercase when they are copied to DOS, but DOS files remain in uppercase when they are copied to a UNIX system. (DOS is not case sensitive).
paths	Paths are separated by a backslash (\), rather than a slash (/).
length	DOS file names are limited to eight characters (called the file name) followed by a period, followed by three characters (called the extension). UNIX files with names which are too long lose the trailing letters.
links	DOS does not recognize links. If you use **doscp** to copy a link to a DOS disk, a complete copy of the file is made. So, if you have two links to the same file called *file1* and *file2*, and copy them both to the same DOS disk, the result will be two identical copies of the file, named *FILE1* and *FILE2*.

You should write-protect your back-up floppy disks to prevent accidental erasure or overwriting. On 5.25-inch floppies, cover the square notch on the side of the disk with the supplied write-protect sticker. On 3.5-inch floppies, slide the write-protect tab open.

Chapter 12

Controlling the work environment

The programs you run define your work environment. The UNIX system is extremely customizable; you can add and remove programs to make your work easier, you can change the priority of jobs (programs that are running), and you can set up the UNIX sysytem so that specific programs run and their required variables are loaded whenever you start a new UNIX session.

Improving performance

The UNIX system is multitasking; many programs may be running simultaneously. You can make the UNIX system operate more efficiently by changing the priority the UNIX system assigns to individual programs, running long programs in the background, and removing programs that are not doing what you expect. But first, before you can do any of these tasks, you need to know what programs are running.

Determining which programs are running

To find out what programs you are running, enter **ps** (process status). This will display information about your current programs, in columns for PID (Process ID; see below), TTY (the terminal on which the command is running), TIME (elapsed time) and COMMAND (the name of the program).

To find out all the programs running on the system that you are authorized to see, enter **ps -ef | more.** This shows all the programs that are running, rather than just your own. It also provides information about the PID of the program's parent, the UID (identity) of its owner, and the current state of the program.

Among the programs that are running, you will see **ps** and **more**. When you entered **ps -ef | more**, you created two new *processes*. (A "process" is a program that is loaded and is running.)

Running programs in the background

You can run a non-interactive program "in the background" (so that while it executes, you can get on with something else) by using the " & " notation. For example, to run **sort** in the background, enter:

```
$ sort file > sorted &
2360
$
```

The number that appears before the prompt is the Process ID (PID) of the **sort** command. If you want to stop the process before it completes, you will need to use this number.

It is not appropriate to run interactive commands such as **vi** in this way.

Continuing programs after logging off

To run a background program that will continue after you log off, enter:

nohup ***program_name*** **&**

nohup means "no hang-up". A program started in this way will continue until it finishes and will not be aborted by your UNIX session's end.

For example, if you are about to print a very long file using the text formatter **nroff**, but need to log out in order to go and do something else, you can enter:

```
nohup nroff myfile > formatted
exit
```

nroff runs in the background and does not stop when you log off. Any error output from the program will be saved in a file called *nohup.out*.

Managing demanding background jobs

To reduce the demands a program makes on the UNIX system, use the **nice** command. For example:

```
nice -20 find / -name something -print > outfile &
```

find runs in the background, sending its output to *outfile*. By using **nice** with a value of "20" to run it, you make the UNIX system spend less time attending to **find** than to any other programs running at the same time.

nice needs a number to tell it how "nice" to be to other users. (A "nice" program is one that does not take over the system.) The number is between 1 and 20; a small value is less "nice" than a large value.

A demanding background process (such as a compiler or text formatter) can slow down the whole system. You can make life easier by reducing the amount of time the UNIX system spends on that program. It will take longer to run, but will not slow down your other tasks.

Stopping runaway processes

If you have started a process running in the background and need to halt it before it finishes, you can use the **kill** command. For example:

```
kill -15 2360
```

kill sends a *signal* to the target process. A signal is a special message with one of several pre-defined values. The first number (after the "-") identifies the signal to send; signal 15 is a command to terminate. The second number is the PID (process ID number) of the process to which to send the signal.

You can obtain the PID of a process using the **ps** command; see "Determining which programs are running" (page 97). If you know the name of the program you want to stop, you can use the following command to find its PID:

```
ps -u login_name | grep program_name | grep -v grep
```

where ***login_name*** is your login name (truncated to seven characters), and ***program_name*** is the name of the program to find. The PID is the number in the second column.

If **kill -15** fails to halt a program that is out of control, try **kill -9** instead. This is more effective, but does not give the program a chance to close any files it may be working on when it receives the signal.

You cannot kill processes belonging to another user or to the system (unless you are the root, or super-user).

Running command sequences

You can run a sequence of more than one command from a single command line. To send several commands, one after another, separate each of them with a semi-colon; for example:

```
ls > list; vi list; sort list
```

This command sequence creates a list of files in a file called *list*, runs the **vi** editor on the list, then sorts it. (Note that you cannot run vi on the data in a pipe.)

If you want to repeat this sequence of commands, you can write them in a file, then make the file execute as a command. This type of file is called a "script" or a "shell script."

For example, to put the preceding command sequence into a file called *myscript* that can be used as a command:

```
echo 'ls > list ; vi list ; sort list' > myscript
chmod +x myscript
```

Now, when you enter **myscript** (or the name you gave to the file), the commands will be executed one after another.

Note that any file of commands you create must have its attributes set to "executable" before you can run it by typing its name. Otherwise, you will see a message like this:

```
myscript: cannot execute
```

For information on attributes, see "Changing access permissions" (page 80).

As an alternative, you can create a short file containing a list of commands; for example:

```
ls > list
vi list
sort list
```

that will be carried out in the same order when the file is executed.

Running scripts with parameters

You can make a file of commands run with different parameters specified on the command line by using *parameter substitution.* This feature lets you run a script using different data files or options for the programs listed in it. For example, suppose you change your file to read like this:

```
ls > $1
vi $1
sort $1
```

$1 refers to the *variable* called " 1 ". This variable is replaced with the first parameter specified on the command line. So if you enter:

```
myscript myfile
```

the word "myfile" is substituted for all occurrences of **$1** in **myscript** as it is executed.

If you want to pass more than one parameter, you can use the variables **$1, $2, $3,** ... to represent the first, second, third (and subsequent) parameters specified on the command line.

If you want to write a script where *all* the parameters, however many there are, are passed to a program, use the variable **$***. For example, a script containing the following:

file $*

will run **file** on every item specified on the command line, whereas:

file $2 $3

will only run **file** on the second and third parameters.

Setting variables

A variable is a label the UNIX system uses to refer to some variable quantity it needs to track. Each variable has a *name* and a *value*, that is stored in the variable.
UNIX variables are known collectively as the *environment*. Many programs use variables to store information temporarily.

To find out what variables are currently set, enter:

env

You will see a long list of information, looking something like this:

```
LANG=english_us.ascii
OALIB=/usr/lib/oa2
HZ=60
MAILOPTS=-dtpubs_lw1
PAGER=/usr/sco/bin/less
VISUAL=/bin/vi
PATH=/bin:/usr/bin:/u/charless/bin:.:/usr/sco/bin:/u/Admin/Scripts
        ...
```

The name on the left of an equals sign is that of a variable; the information on the right is the value associated with the variable. For example, **LANG** is a variable name, and **english_us.ascii** is its value.

To set a new variable, enter:

myvariable=*value*

where ***value*** is whatever you wish **myvariable** to equal. (If you are using the C shell instead of the Bourne or Korn shells, you will need to enter **setenv myvariable *value*** instead.)

If you want to make **myvariable** accessible to all programs you run, follow it with the command **export myvariable**. (This is not necessary in the C shell.)

For example, suppose you want to make the shell load and run programs stored in a directory called */u/myprogs/bin*. When you enter the name of a program, the shell looks for it in those directories that are listed, separated by colons, in the variable called **PATH**. To add */u/myprogs/bin* to your **PATH** variable, enter:

```
PATH=$PATH:/u/myprogs/bin
```

This replaces the current value of **PATH** with itself, followed by **:/u/myprogs/bin**.

To make this available to all programs you run in the remainder of this UNIX session, enter:

```
export PATH
```

Removing variables

To remove a variable, enter:

unset ***variable***

where ***variable*** is the name of the item to remove. (In the C shell, enter **setenv** ***variable*** **'""'** to remove ***variable***).

Referring to variables

You can refer to the contents of a variable within one of your scripts. To do so, insert the name of the variable, preceeded by a " $ " symbol. The value of the variable will then be substituted. For example, to save your current directory in a variable, create a file containing the following:

```
current=`pwd`
```

This command assigns the output from running **pwd** to the variable **current** when you run it.

If you **cd** to another directory, you can return to this one by typing:

```
cd $current
```

from wherever you are.

For example, the variable **$HOME** contains the path to your home directory. Suppose you want to find files located in one of your subdirectories. You can use a script like this:

```
find $HOME -name $1 -print
```

Supposing the script is called *whereis*, and you want to find a file called *new.document*, you would enter **whereis new.document**. **whereis** will only search those directories that are located within your home directory.

If you give this script to a friend, it will do the same for that user — searching only the files below that person's home directory — because the value of **$HOME** is unique to each user.

Changing your password

To change your password from the UNIX command-line, enter **passwd**.

passwd will ask you to type in your old password before you select a new one. If you choose to select a password of your own, **passwd** will ask you to enter it twice, and will make sure that it is not a word that is easy to guess. If you do not choose a password, **passwd** will generate a suitable password for you.

You should take care not to write your password down anywhere, not to tell it to anyone, and to choose one that is difficult to guess (avoid names, places, or telephone numbers). For further information about choosing a password, please see the *System Administrator's Guide* or the **passwd** manual page.

Note that if you forget your password you will not be able to log on again. You will have to ask your system administrator for help.

Setting up environment variables

It is possible to set environment variables automatically when you log on. Whenever you log on, your shell looks for a special login command file (or startup file) called either *.profile* or *.login* and executes any commands it finds in it.

To find your login command file, enter:

```
ls -a
```

There may be more than one file in the listing. If you are using the C shell, you need to edit *.login*. If you use the Korn shell or the Bourne shell you must edit *.profile*.

To add a new variable to your startup file, simply edit the file and insert two lines like:

```
my_variable=Seven
EXPORT my_variable
```

(Note that this does not apply to the C shell.) This will set the value of **my_variable** to "Seven" and *export* it so that sub-shells can make use of it. (An unexported variable is only available to the shell within which it is defined.)

A sub-shell is a shell run from within another shell. For example, when you execute another command from within the **vi** editor, it is running in a sub-shell. For more information on shells, see "Changing your shell" (this page).

Changing your shell

If you want to change your default shell (the one you use when you start work), you should ask your system administrator. Information on how to change the default shell is provided in the *System Administrator's Guide.*

The shell is the program that the UNIX system uses to communicate with you, the user. The shell reads your instructions and carries them out, locating and running programs and interpreting scripts. (It is called the shell because it puts a shell around the core of the UNIX system, making it easier to work with.)

The following shells are available:

Korn shell	The shell of choice, it provides facilities for recalling and editing commands you have already typed in, and for controlling background programs.
Bourne shell	The original UNIX shell, it predates the Korn shell and offers fewer facilities.
C shell	This shell has a C-like syntax with basic command recall facilities. It is incompatible with the Korn and Bourne shells.
SCO shell	This is a menu based shell, designed to help inexperienced users master the complexities of the UNIX system without recourse to the other, command-line shells.

Remember, if you request a change of shell, you must move any environment variables you have set up into the appropriate file in order for them to be set when you start a UNIX session. The files to check are *.cshrc* and *.login* for the C shell, *.profile* for the Bourne shell, or *.kshrc* and *.profile* for the Korn shell.

See the introduction to this guide for sources of more information about shells.

Chapter 13
Editing files

The UNIX system includes several editors, each optimized for specific needs:

vi	This visual editor is used for interactively writing and editing files of text. **vi** is the editor you will use most in the UNIX window. It resembles the Desktop editor to some extent, but provides no menus or help. (There is a variant of **vi** called **vedit** that is set up for novices, and a variant called **ex** that behaves like **ed**, below.)
ed	The original UNIX line editor, **ed** is *line-oriented*; it can only edit a line at a time. **ed** is used within shell scripts, and when it is impossible to configure a terminal properly.
sed	The stream editor, **sed** reads its input file, carries out a sequence of commands, and writes the result to its output file. **sed** cannot be used interactively.

The UNIX system also provides a variety of tools for formatting, spell-checking and processing text. See Chapter 10 (page 77) for information on some of the programs available.

Using vi

vi is the standard UNIX tool for editing text. It differs from the Desktop editors in that, instead of being controlled through menus, it is controlled exclusively by commands you type into it.

Because it is designed to read commands from a variety of terminals, **vi** has two *modes*: insertion mode and command mode. If you are new to **vi**, you may find this confusing at first. Try to remember this simple rule: you cannot issue commands when in insertion mode (except for the command to switch to command mode), and you cannot enter text while in command mode.

Starting vi

To start editing a file, enter:

vi *filename*

If the file already exists, vi will read it in. If it does not exist, vi will create it.

When you start **vi**, you are in command mode. **vi** has two modes; command mode, and insertion mode. In command mode you can issue commands to **vi** and move around your document. In insertion mode, you can only enter text.

Stopping vi

To leave **vi** you must switch to command mode, if you are not already in it. You can enter command mode by pressing ⟨Esc⟩. The terminal beeps or flashes at you if you are already in command mode and press ⟨Esc⟩.

There are several ways to leave vi. Here are the most common:

ZZ	Save the current file and exit. (Just type a capital "Z" twice.) This command will not work if the current file is write-only, or you are attempting to edit more than one file. (This command is equivalent to **:w :q** or **:wq**.)
:w	Save the current file (**w** is short for write file). Do not exit. This command will fail if the file is write-only. You can save under a different name by adding a filename: for example, **:w newfile** saves the current file as *newfile*. (Note that the colon (:) tells **vi** to read everything you type until the next ⟨Enter⟩ as a single command.)
:q	Quit **vi**. This command will fail if the file has changed since the last time you saved it. (If you really want to quit without saving, enter command mode and type **:q!**. This causes **vi** to quit without saving any changes you have made to the current file.)
:!*cmd*	Execute the program *cmd*, then return to **vi**. The command **:!sh** is *not* the same as exiting **vi**; **vi** is still running, and when you exit the shell (by typing exit or ⟨Ctrl⟩D) you will return to **vi**.

What to do if you encounter trouble

First press ⟨Esc⟩ twice. If a command is in progress, the ⟨Esc⟩ key cancels it. If you are in insertion mode, the ⟨Esc⟩ key puts you back into command mode. If your terminal beeps or flashes when you press ⟨Esc⟩, it means you are now in command mode.

If the UNIX window is unreadable, press ⟨Ctrl⟩L in command mode. **vi** then redraws (refreshes) the window.

If you still cannot read the UNIX window, either your terminal is set up incorrectly or you are editing a non-text file. Type **:q!** to exit without saving the current file.

Entering text

To type text into a file, you must switch from command mode to insertion mode.

To enter insertion mode, press **i** (for insert).

To leave insertion mode, press ⟨Esc⟩. The terminal will beep or flash if you press ⟨Esc⟩ again.

If you are not sure which mode you are in, press ⟨Esc⟩ until the terminal beeps or flashes. You will then be in command mode.

When you are in insertion mode, anything you type is entered into the document at a position immediately behind the cursor. If you make a typing mistake, you can use the ⟨Bksp⟩ key to backspace over the error. When you have finished inserting text, press ⟨Esc⟩ to return to command mode.

Moving around inside the file

You must be in command mode before you can move the cursor around the file. If you are not already in command mode, you can enter it by pressing ⟨Esc⟩.

To move a single character width in any direction, use the arrow keys on your keyboard. (The keyboard keys " h "," j "," k ", and " l " also move the cursor.)

You can move around in various units:

word	To move forward or backward a word at a time, press the " w " (word forward) or " b " (backward) key.
start/end	To move to the start of a line or the end of a line, press the " ^ " (start) or " $ " (end) key.
sentence	To move forward or backward a sentence at a time, press the ") " (next sentence) or " (" (previous sentence) key.
paragraph	To move forward or backward by a paragraph, press the " { " (previous paragraph) or " } " (next paragraph) key.

window	To move forward or backward by a window full of text, press the ⟨Ctrl⟩**F** (forward) or ⟨Ctrl⟩**B** (backward) key or the ⟨PgDn⟩ or ⟨PrevPg⟩ key.
line number	" G " (goto). **G** without a number takes you to the last line in the file. If you enter **1G** you will go to the start of the file.

To see your current line number, press ⟨Ctrl⟩G. A status line will appear at the bottom of the UNIX window, telling you the name of the file, whether it has been modified, your current line number, the number of lines in the file, and your position in the file as a percentage of the length of the file.

To make any of these commands repeat, enter a number (the number of times you wish the command to repeat), then the command. For example, to move forward five words, enter **5w**.

Deleting text

You must be in command mode before you can delete or change text.

To delete text, use the **d** command followed by the unit of text to delete. Options are:

dl	delete letter (or type **x** — a shortcut)
dw	delete word
dd	delete line

To delete several units of text at a time, enter the number of units to delete, followed by the appropriate command. For example, to delete five words, type **5dw**.

To delete a range of lines, enter a command in the form:

:*x*,*y*d

where *x* and *y* are the first and last line numbers in the range to be deleted. For example, to delete lines seven through seventeen inclusive, use the command:

:7,17d

Replacing text

To replace a single letter with another letter, position the cursor over the letter and type the **r** command, followed by the replacement letter.

To replace an unlimited amount of text on the current line with new text, position the cursor over the first letter and type **R**. You are now in replace mode. This corresponds to insertion mode, but characters you type will replace the previously existing text. You can return to command mode by typing ⟨Esc⟩.

Inserting text

To insert text at the start of a line, regardless of where the cursor is within the line, type the **I** command.

To add text to a line, type the **a** command. This puts you into insertion mode, but text is added after the current cursor position.

To add text to the end of a line, regardless of where the cursor is in the line, use the **A** command.

Modifying text

To change the case of text (from uppercase to lowercase or vice versa), use the **~** command. Place the cursor over the text to change and press "~" once for each character. (The cursor advances by one character each time you issue this command.)

To swap two characters, position the cursor over the left character and type **xp**. The two characters will be transposed.

To open up a line below the line the cursor is on, type the **o** command.

To open up a line above the line the cursor is on, type the **O** command.

To join together two lines, type the command **J**. The line below the cursor will be joined onto the end of the line the cursor is on.

To undo the last command, type the command **u**. The result of your last command will be undone.

To undo all changes to the current line since you last moved the cursor to it, type the command **U**.

Cutting and pasting text using buffers

A buffer is where **vi** temporarily stores text. **vi** has twenty-six buffers, named " a " through " z ".

To copy a line of text into a buffer, type the command

> **"*buffer_name*yy**

(the **yy** command is short for yank). To copy several lines, precede the command with the number of lines you wish to copy; for example, to copy fifteen lines into buffer " a ", type **"a15yy**.

To delete a line of text, saving it in a buffer, use the command **dd** instead of **yy**.

To paste the contents of a buffer into the text immediately above the cursor, type the command **"*buffer_name*P**. For example, to paste the contents of buffer " g " into your file above the cursor, type **"gP**. The paste command **p** (lowercase " p ") pastes the buffer in below the current line instead.

You can cut or copy a region of text of any size with the *marker* facility. Place a marker in the text at the beginning of the region you want to move, by typing **m** followed by a letter (" a " through " z "). This inserts an invisible marker at your current position. Now move to the end of the region. To *copy* the text between the cursor position and the mark into buffer " a ", type **"ay`a**. To *move* the text between the marker and the current cursor position into buffer " a ", type **"ad`a**.

The command is built up as follows. First, specify that you are going to use a buffer by typing **"a** (for buffer " a "; you can use any other buffer you like). Second, specify whether you are going to cut text (" d " for delete) or copy text (" y " for yank) into the buffer. Third, use the **`*marker*** command to cut or copy the text between the current cursor position and the named marker. (**`*marker*** means "go to ***marker***" and ***marker*** stands for the mark you placed in the text.

Searching for text

You can search for text if you are in command mode. To search forward, type ***/text***, where ***text*** is the text you want to find. If you find it, you can repeat the search for the next occurrence of the text by typing " / " or " n ".

To search backward, type " ? " instead of " / ".

vi searches for text using wildcards, as does **grep**. The following wildcards are used:

.	(period) matches any single character except a newline. For example, **/g..d** matches "good" but not "god".
*	(asterisk) matches one or more instances of the last character specified. For example, **/f*** matches any number of f's, while **/.*** matches any number of any character.
^	(caret) matches the start of a line
$	(dollar) matches the end of a line
[..]	matches a set of characters. Any of the characters within the square brackets will be recognized.
\	(backslash) takes away the special meaning of the character to the right of the backslash. For example, **.*** matches any number of any character, but **\*** matches a single asterisk.

Substituting text

To substitute one sequence of characters for another on the current line, use the **:s/*old*/*new*/** command, where ***old*** is the sequence to find, and ***new*** is the sequence that replaces it.

To substitute all occurrences of a sequence of characters within a file, type:

:g/*old*/s//*new*/g

You can search for wildcards, but should replace them with a string of ordinary text. For example, to search for any word beginning with "cent" (such as center, centered, or central) and replace it with "middle" instead, type:

:g/cent.*\ /s//middle/g

Configuring vi

vi has a number of internal variables that can be configured with the **:set *varname*** command, where ***varname*** is the name of the variable to change.

To examine the state of **vi**'s settings, go to command mode and type **:set all**. You will see a list of settings.

If a variable name starts with "no", it is not set (that is, not switched on). You can set it by typing **set *varname*.** If a variable name does not start with "no", and is not followed by a number, it is set. For example, if you want to make **vi** ignore wildcards, you must switch off the variable **magic**. To do this, type **set nomagic**. If the variable name is followed by a number, you can change its value by typing **set *varname=value*,** where ***value*** is the new setting you want it to have.

To make **vi** automatically begin a new line before you reach the right side of the UNIX window, type **:set wrapmargin=15**. This makes **vi** "wrap" the first word you begin to type that is less than fifteen characters from the right side of the window. (If you have used other word processors, this feature may be familiar to you as "word wrap.")

A complete list of the internal **vi** variables and their meanings is included in the **vi** manual page.

Summary of vi commands

The following tables contain all the basic **vi** commands and variables. Complex commands are omitted; see the **vi** manual page for details.

Table 13-1 Entering vi

Typing this:	does this:
vi ***file***	starts at line one
vi +n ***file***	starts at line ***n***
vi + ***file***	starts at last line
vi ***+/pattern file***	starts at ***pattern***
vi -r ***file***	recovers ***file*** after a system crash

Table 13-2 Cursor movement (command mode)

Pressing this key:	does this:
h	moves one space left
l	moves one space right
⟨Space⟩	moves one space right
w	moves one word right
b	moves one word left
k	moves one line up
j	moves one line down
⟨Enter⟩	moves one line down
)	moves to end of sentence
(	moves to beginning of sentence
}	moves to beginning of paragraph
{	moves to end of paragraph
⟨Ctrl⟩**U**	scrolls up half a window
⟨Ctrl⟩**D**	scrolls down half a window
⟨Ctrl⟩**F**	scrolls down one whole window
⟨Ctrl⟩**B**	scrolls up one whole window

Table 13-3 Inserting text (command mode, enters insertion mode)

Pressing this key:	starts insertion:
i	before the cursor
I	before first character on the line
a	after the cursor
A	after last character on the line
o	on next line down
O	on the line above
r	on current character, replaces one character only
R	on current character, replaces until ⟨Esc⟩

Table 13-4 Delete commands (command mode)

Command	Function
dw	deletes a word
d0	deletes to beginning of line
d$	deletes to end of line
3dw	deletes three words
dd	deletes the current line
5dd	deletes five lines
x	deletes a character
5x	deletes five characters

Table 13-5 Change commands (command mode, enters insertion mode)

Command	Function
cw	changes one word
3cw	changes three words
cc	changes current line
5cc	changes five lines

Table 13-6 Search commands (command mode)

Command	Function	Example
/and	finds the next occurrence of *and*	and, stand, grand
?and	finds the previous occurrence of *and*	and, stand, grand
/^The	finds next line that starts with *The*	The, Then, There
/[bB]ox/	finds the next occurrence of *box* or *Box*	
n	repeats the most recent search, in the same direction	

Table 13-7 Search and replace commands (command mode)

Command	Result	Example
:s/pear/peach/g	all *pears* become *peach* on the current line	*pear* becomes *peach* if present on the current line
:1,$s/file/directory	replaces first instance of *file* on each line with *directory* from line 1 to the end	*filename* becomes *directoryname*
:g/one/s//1/g	replaces every occurrence of *one* with 1	*one* becomes *1*, *oneself* becomes *1self*, *someone* becomes *some1*

Table 13-8 Pattern matching: special characters (regular expressions)

This character:	Matches:
^	beginning of a line
$	end of a line
.	any single character
[...]	a set of characters (represented by "..." ; the range can either be specified, like **[abc]** or a range like **[a-b]**)

Table 13-9 Leaving vi (command mode)

Command	Result
:w	writes out the file
:x	writes out the file, quits **vi**
:wq	writes out the file, quits **vi** (like **:w :q**)
:q!	quits **vi** without saving changes
:!*command*	executes UNIX ***command***
:sh	starts a new shell
!!*command*	executes ***command*** and places output on current line
:e *file*	edits ***file*** (save current file with **:w** first)

Table 13-10 Options (:set option)

This option:	does this:
all	lists all options
term type	sets terminal to type
ignorecase	ignores case in searches (on or off)
list	displays ⟨Tab⟩ and end-of-line characters (on or off)
number	displays line numbers (on or off)
report	prints number of lines changed by a line-oriented command
terse	shortens error messages (on or off)
warn	turns off "no write" warning before escape (on or off)
magic	allows inclusion of special characters in search patterns without a preceding backslash (on or off)
wrapscan	prevents searches from wrapping around the end or beginning of a file (on or off)
mesg	permits display of messages sent to your terminal with the **write** command (on or off)

Using ed

ed is a line editor; it edits a single line at a time. It makes no use of the terminal, and the cursor movement keys associated with **vi** have no effect.

Instead of showing you a window of text, **ed** relies on *line addresses.* A line address is the number of a line in a file, to which a command is applied. Commands in **ed** may all have zero, one- or two-line addresses. (In the latter case, the two addresses correspond to a range of lines within which the command is carried out.)

Starting ed

From the shell prompt, type **ed** ***filename***, where ***filename*** is the file to edit. **ed** starts, then displays the number of lines it has read from ***filename***. You are in command mode.

Leaving ed

ed uses **vi**-like commands (of the type prefixed by a colon (:)). (**vi** is descended from **ed**.) To quit **ed**, use most of the **vi** commands except **ZZ**. Commands are not prefixed with a colon. For more information, see "Stopping vi" (page 106) or the **ed** manual page.

Reading a file

To see the contents of your file, use the **l** (list) command. **list** requires line addresses. If no addresses are given, **ed** just displays the current line.

To see lines one to ten of a file, type:

```
1,10l
```

The first digit is the start address, and the second digit is the stop address for the command l (list). **ed** displays everything from the start address to the stop address.

To refer to the current line, use " . " (a period); this is the address of the current line.

To refer to the last line of the file, type " $ ".

You can use *relative* addresses; for example, **$-5** means the fifth line before the last line of the file, while **.+2** means the second line after the current line.

For convenience, a comma (,) stands for the address pair **1,$** (that is, the entire file), while a semicolon (;) stands for the address pair **.,$** (current line to end of file). For example, to list the entire file, type **,l**.

Editing text

There are several commands for editing text in **ed**:

c	Change text: this command uses an address. The specified lines are deleted, then whatever you type replaces them. When you have finished entering text, press ⟨Ctrl⟩**D** to return to command mode.
i	Insert text: this command uses an address. Whatever you type is inserted before the specified line; press ⟨Ctrl⟩**D** to stop inserting text.
d	Delete text: this command uses an address. The lines you specify are deleted.
s/*old*/*new*	Replace text: this command searches each addressed line for ***old***, and the first occurrence is replaced with ***new***. ***old*** can be a wildcard, as with **vi**. If you supply a range of addresses to this command, each line in the range is searched and the first instance of ***old*** on each line is replaced with ***new***.

ed supports most of the same wildcards as **vi**. For full details, see the **ed** manual page.

Using sed

sed is a stream editor; it cannot be used interactively like vi or ed. Commands to sed are entered in a script file. sed reads a line of input, executes all the applicable commands in its script, and writes the result to its output. It repeats this cycle until there is no more input.

Running sed

To start sed, use the following command:

```
sed -f script_name < input > output
```

where *script_name* is the name of a file containing a script of instructions, and *input* and *output* are the input and output files.

sed will execute the script until there is no more input. It will then exit.

If you want sed to execute a short command on the contents of a file, you can enter the commands on the command line using the -e option:

```
sed -e 'sed commands' input > output
```

sed commands

sed commands are similar to those of ed. However, sed does not allow relative line addresses, because sed never backs up; it reads through the input file just once, from start to finish.

For a full description of the sed commands, see the sed manual page.

Chapter 14

Getting started with DOS

In all important respects, using SCO Open Desktop/SCO Open Server DOS Services is just like using DOS on a stand-alone personal computer. With SCO Open Desktop/SCO Open Server, you can:

- use all common DOS commands.
- install and run off-the-shelf DOS applications.
- use your computer hardware (diskette drives and printers, for example) in standard DOS ways.

In addition, with DOS Services, you can:

- Run Microsoft Windows 3.0 and 3.1 (Windows 3.1 runs in the "standard" or 80286 "protected" mode)
- run several DOS applications in separate DOS environments simultaneously and switch between any of the DOS and UNIX windows.
- take advantage of the security capabilities of the UNIX system, including password protection for the whole system and protection for specified DOS directories, data files, and programs.
- access UNIX data files and programs, including files within a network environment.

Running a DOS command or application directly from your Desktop or UNIX shell works because SCO systems automatically distinguish DOS executable files from UNIX executable files.

This is possible because DOS Services creates a *virtual personal computer* (also called a *virtual PC* or *virtual machine*) for you whenever you run DOS. Because DOS Services virtual computers use the virtual 8086 mode of your 80386 or 80486 processor, you can run DOS commands and applications under DOS Services only if they are compatible with the Intel 8086 processor, except for

Windows 3.1, which is supported in the "standard" or 80286 "protected" mode. DOS Services does not support any other DOS applications that require the protected mode available on 80286, 80386, and 80486 processors.

Beginning a DOS session

You can enter the DOS environment from the Desktop in one of three ways. You can:

1. Double-click on the DOS icon from the Desktop. (The DOS icon is located in the Accessories window, which can be opened by double-clicking on the Accessories icon.)

   -or-

2. Double-click on the UNIX icon to open a UNIX window, and then type **dos** at the UNIX prompt.

   -or-

3. Double-click on an icon that represents a DOS program (such as Microsoft Windows).

If you use either method 1 or method 2, the DOS window opens with the the standard DOS prompt displayed:

```
C:
```

If you use method 3, the DOS window opens with the the selected program already running. For example, double-clicking on the Lotus 1-2-3 icon causes Lotus 1-2-3 to run without first displaying a `C:` prompt.

The `C:` prompt that you see after invoking DOS with either of the first two methods tells you that you are using DOS drive C: (the fixed disk). You can now use your computer as you would use a standard computer running DOS.

See "Controlling the DOS window" (page 142) for more about the DOS window.

Ending a DOS session

To end a DOS session and return to the Desktop, type (at your DOS prompt):

quit

Or, if you are running a DOS executable file (such as Lotus 1-2-3), simply exit the program. The Desktop reappears, and you can continue to use SCO Open Desktop/SCO Open Server.

Using DOS commands and applications

With some exceptions (see "Inapplicable DOS commands") all common DOS commands work as they do on a conventional, stand-alone DOS computer.

You can use DOS commands for routine operations like copying files or listing the names of files on the fixed disk or a diskette. There are also DOS commands for more specialized purposes such as creating text files and creating and executing BASIC programs. These DOS commands are all supplied with DOS Services and are described in your DOS documentation. You can also install and use off-the-shelf DOS applications in the DOS environment. *Administering DOS Services* in the *System Administrator's Guide* as well as Controlling the DOS work environment (page 141) and "Installing Windows applications" (page 199) provide instructions for installing DOS applications for use on your SCO system.

In the DOS environment, you specify directories and give options to commands in the usual DOS way. If you are using the *C:* drive, the following command displays the contents of the directory *\USR\DBIN* in wide format, with five files listed per line:

```
dir \usr\dbin /w
```

If you are a UNIX user who has not used DOS, you may be unfamiliar with the use of the slash (/) to turn on options and the backslash (\) as the path separator. For further information on this syntax, refer to your DOS documentation.

Changing the default drive: When you enter the DOS environment, your default drive is drive *C:* (the fixed disk) and your prompt is `C:\`. To change your default drive to drive *A:*, be sure you have a valid, formatted DOS diskette in the drive, and type:

```
a:
```

Your prompt changes to `A:\` and you can execute commands from the diskette drive.

If your system has a second diskette drive, you can use it with DOS by referring to it as drive *B:*. The diskette drives are available to only one user at a time. If one user is accessing a diskette drive and a second user attempts to use it at the same time, the second user sees a message stating that the drive is unavailable.

Changing directories: Use the DOS **CD** or **CHDIR** command to change your current working directory. To change to *C:\USR\DBIN*, for example, type:

```
cd \usr\dbin
```

Piping and redirecting with DOS: Pipes and redirection function in DOS Services as under standard DOS:

```
dir | sort > contents
dir a: >> contents
```

All common DOS commands work as you would expect in the DOS Services environment, including **COPY**, for copying files; **COMP**, for comparing files; **TYPE**, for displaying the contents of files; **REN**, for renaming files; and **DEL**, for deleting files.

The more specialized DOS tools for editing files, programming, and configuring the DOS environment also work in the DOS Services environment. These tools include:

- the **EDLIN** and **EDIT** editors, and the **QBASIC** interpreter,
- batch files, including all standard batch commands,
- the **DEBUG** utility, and
- *CONFIG.SYS* files.

Using off-the-shelf DOS applications

You can use most off-the-shelf DOS application programs in the DOS Services environment just as you would use them on a stand-alone DOS personal computer. You can also use custom DOS applications that you might have developed.

To use an application from drive *A:*, follow the application manufacturer's instructions. Typically, you insert the application diskette into drive *A:*, change your current drive to drive *A:*, and invoke the application by name from your A> prompt. For example:

```
a:
wp
```

To run an application from drive C:, you must first install the application on the fixed disk.[1] Once installed, applications are executed according to the manufacturer's instructions. For example, if WordPerfect is installed on your fixed disk, you can start it by typing:

```
wp
```

1. In most cases, DOS applications are installed by following the application manufacturer's instructions. For further pointers on installing DOS applications, see *Administering DOS Services* in the *System Administrator's Guide.*

Booting applications from drive A:

A few personal computer applications (such as some versions of the Microsoft Flight Simulator®) must be booted from drive *A:* because they do not run under DOS. (They actually boot their own operating systems.) To use these applications on a conventional personal computer, you insert the bootable application diskette into drive *A:* and power the computer on or press ⟨Ctrl⟩⟨Alt⟩⟨Del⟩.

To run these applications on DOS Services, you use the **dosboot** command. To use **dosboot**, you must be using the UNIX shell and not the DOS environment. If you are currently in the DOS environment, type **quit**. Your prompt should be `$` or `%`.

Only two steps are required to use **dosboot**:

1. Insert your bootable application diskette into drive *A:*
2. Type:

```
$ dosboot
```

When you use **dosboot**, your application runs independently of any other UNIX or DOS activity. This means that files on drive *C:* are not available, and you cannot type **quit** as you usually do to leave the DOS environment.

To end a **dosboot** session, press ⟨Ctrl⟩⟨Alt⟩⟨Del⟩.[2] Your UNIX system prompt then returns.

File permission errors

Sometimes the message DOS returns is affected by file permission modes. For example, when a DOS command you issue encounters a file for which you do not have read access, DOS may display a message that implies the file does not exist, even though the file does exist. Similarly, if you try to create a file in a directory for which you do not have write access, DOS may display an error message such as `File creation error` that does not clearly indicate the nature of the problem.

2. Note that you can also use the KILL DOS control code described in "Stopping DOS programs" (page 125) if your application is hung and does not respond to ⟨Ctrl⟩⟨Alt⟩⟨Del⟩.

Inapplicable DOS commands

Nearly all standard DOS commands operate in the DOS Services environment just as they do on a conventional stand-alone DOS computer. Some DOS commands, however, are either not usable in the DOS Services environment or operate differently than they do on a stand-alone DOS computer.

In particular, some of them will operate correctly only on a "real" DOS filesystem. DOS filesystems are indexed by a File Allocation Table. DOS Services emulates DOS filesystems while preserving the underlying UNIX structure, which is completely different.

The following restrictions apply:

- You cannot use the DOS **FDISK** command under DOS Services. Instead of running **FDISK** under DOS Services, use equivalent UNIX utilities or shut down the UNIX system, boot standard DOS, and use **FDISK** under standard DOS.
- You cannot use **SHIP** or any other DOS command for parking the fixed disk head on the DOS Services system.
- You cannot use the following commands on the shared UNIX/DOS filesystem: **CHKDSK, FORMAT, SYS, MIRROR,** or **UNFORMAT**. Do not use them on Drives *C:*, *D:* or *J:*.

  You *can* use these commands on a real DOS filesystem, such as the diskette drive, or a physical DOS partition. You can also use them or virtual floppies and virtual DOS partitions, because, though these are portions of the shared UNIX/DOS filesystem, they are formatted as real DOS filesystems.

  FORMAT may work somewhat differently under DOS Services and standard DOS. A filesystem which you **FORMAT** under DOS Services may work properly under DOS Services but not work properly under raw DOS. It is safest to use raw DOS to **FORMAT** any disk or partition from which you intend to boot DOS. They will then work properly under either DOS or DOS Services.

  Since virtual floppies or virtual partitions will be used only under DOS Services, they should be **FORMAT**ed or **UNFORMAT**ed under DOS Services.
- Similarly, you cannot use **UNDELETE** on any file which is part of the shared UNIX/DOS filesystem. But you *can* use it on any file which is part of a real DOS filesystem. Use **UNDELETE** on real or virtual floppies or on real or virtual DOS partitions.

- You *can* use the DOS **TIME** and **DATE** commands to display or change the time and date that apply to the DOS environment, but when you leave the DOS environment, time and date are determined by the UNIX clock. When you reenter DOS, the DOS clock is always initially synchronized with the UNIX clock.

If you issue a DOS command that does not work in the DOS Services environment, DOS displays an error message but does not harm your computer in any way or destroy any data.

Stopping DOS programs

There are several ways to stop DOS programs that you start in a DOS environment. Most DOS applications include a specific procedure for stopping their execution. Whenever possible, you should stop a DOS program using the procedure designed for that program. Sometimes, however, you might want to stop a DOS utility that provides no specific method for termination, or else a DOS application might get locked into a state where the prescribed termination procedure does not work. If you run into one of those conditions, follow one of these procedures:

1. Use the DOS break character ⟨Ctrl⟩⟨C⟩ or ⟨Ctrl⟩⟨Break⟩ just as you would in standard DOS. These functions stop DOS commands like **DIR**, **TYPE**, or **TREE**, and some applications. When you press ⟨Ctrl⟩⟨C⟩, your DOS prompt returns and you can resume DOS work immediately.

2. If ⟨Ctrl⟩⟨Break⟩ does not work, press ⟨Ctrl⟩⟨Alt⟩⟨Del⟩. That is, press ⟨Ctrl⟩ and ⟨Alt⟩ at the same time; then, while still holding ⟨Ctrl⟩ and ⟨Alt⟩, press ⟨Del⟩. This is the key sequence used to reboot DOS on a standard DOS computer. In DOS Services, ⟨Ctrl⟩⟨Alt⟩⟨Del⟩ causes the DOS program as well as the current DOS environment to abort. You must reinvoke DOS before you can resume DOS work.

   WARNING When you press ⟨Ctrl⟩⟨Alt⟩⟨Del⟩, you could lose data if your DOS program is working on open files, just as you would on any standard DOS system.

3. If neither the ⟨Break⟩ character nor ⟨Ctrl⟩⟨Alt⟩⟨Del⟩ properly terminates your DOS process, use the **KILL** DOS control code appropriate for your terminal.

DOS Window or Console	PC Scancode Terminal	ASCII Terminal
⟨Ctrl⟩⟨Esc⟩ ⟨Ctrl⟩⟨K⟩	⟨Ctrl⟩⟨Esc⟩ ⟨Ctrl⟩⟨K⟩	⟨Esc⟩ ⟨Ctrl⟩⟨K⟩

Chapter 15

Finding your way around with DOS

With SCO system DOS Services, the entire UNIX filesystem is available to you. DOS treats it as a DOS fixed disk, usually referenced as Drive C:.

When you boot DOS on a conventional stand-alone personal computer, your working directory is the root of the filesystem tree. You own all files in the filesystem and can access them easily with **CD** (change directory) commands. You can also modify any file as you please.

On SCO systems, each user has a home directory, that is, a directory containing the files and subdirectories created and owned by that user. When you log in to the UNIX environment on your SCO system and then immediately enter the DOS environment, your working directory is your home directory. (If you change directories before entering the DOS environment, however, your working DOS directory is the same as your UNIX directory at the time you type **dos**.) You can access your own files and subdirectories like you can on a conventional DOS computer.

NOTE Due to differences in the way DOS and UNIX do file allocation, some DOS commands (such as **mirror**((CMD)) do not work on shared filesystems.

The DOS search path

When you run a DOS program by typing a path name, DOS looks in the directory you specify for the program. If the program is there, DOS runs it. If the program is not there, the operation fails. For example, if you enter the following command:

```
C:\usr\ldbin\wp5\wp
```

DOS looks in the directory *\USR\LDBIN\WP5* on drive C: for the program **WP5** and runs it only if it is there.

If you type only the name of the program (for example, **wp**) without specifying its path, DOS looks first in your current working directory for the program. If the program is there, DOS runs it. If the program is not there, DOS searches through the directories in your search path to find the program.

The DOS search path in DOS Services works like the search path on a conventional DOS system, with one difference: when you enter the DOS environment, your search path is automatically set to be the same as your UNIX search path. This path includes the directories *\USR\DBIN* and *\USR\LDBIN*, the directories where standard DOS commands and applications are stored. You can, if necessary, override the default DOS search path by using the **PATH** command as you would on a conventional DOS system. Note that the path is often set in *AUTOEXEC.BAT*. For more information on the *AUTOEXEC.BAT* file, see "Using AUTOEXEC.BAT and CONFIG.SYS files" (page 145).

Naming DOS files and directories

When you create files or directories during a DOS session or using a DOS application, your names must conform to standard DOS rules for length and character set.

You can type the name with either uppercase or lowercase alphabetic characters. When you create a file on a DOS medium (a diskette in drive *A:*, for example), DOS converts all alphabetic characters to uppercase as expected. When you use DOS to create a file in the shared UNIX/DOS filesystem (DOS drive C:), however, DOS Services converts all alphabetic characters to lowercase. Using lowercase for filenames is conventional under the UNIX system. Requiring names to be consistently lowercase also prevents you from creating names that are identical except for case, which DOS cannot differentiate.

Thus, any file you create with DOS Services in the shared UNIX/DOS filesystem can be accessed by either DOS or the UNIX system.

Differences between DOS and UNIX filenames

DOS and UNIX rules for naming files and directories differ with respect to case, size, and character set.

UNIX is case-sensitive while DOS is not. Alphabetic characters in UNIX file- and directory names are usually lowercase, but they can be any combination of upper- and lowercase. Whatever combination you enter is preserved. DOS, on the other hand, interprets all alphabetic characters in file- and directory names as uppercase, whether you enter them in uppercase or lowercase. To the UNIX system, "chapter1" and "CHAPTER1" name two different files; DOS cannot distinguish between the two forms, seeing them instead as the same name.

DOS limits file and directory names to eight characters plus an optional extension of up to three characters. Traditionally, the UNIX system allows names up to 14 characters, although some newer systems (including Open Desktop 2.0) allow more. Although the UNIX system does not provide for filename extensions in the same sense as DOS, a UNIX name can contain a period anywhere in the name. Thus, while the UNIX system accepts any legal DOS name, DOS does not allow such perfectly good UNIX names as *messagetoall* or *chapter.seven*.

DOS and the UNIX system accept both alphabetic and nonalphabetic characters in file and directory names, but the UNIX system accepts more nonalphabetic characters than DOS. For example, control characters and spaces are valid characters in UNIX names but not in DOS names. (Note that UNIX names containing spaces must be enclosed in quotes.)

Accessing files with illegal DOS names

You can use DOS to access any file or directory in the shared UNIX/DOS filesystem, whether it was created with DOS or under the UNIX system. However, you must use a special mapped name for UNIX files or directories with names that do not conform to DOS rules. These names include:

- names longer than DOS allows.
- names with more than three characters following a period.
- names with nonalphabetic characters that DOS does not recognize.
- names with uppercase alphabetic characters.

When any DOS utility or application accesses a UNIX name that does not conform to DOS rules, DOS Services translates, or maps, the name to a legal DOS name by appending a unique index consisting of an apostrophe followed by one to three characters. If necessary, the UNIX filename is truncated before appending the index. For example, a file called *messagetoall* might be mapped to the name *mess'baq*. You can determine the mapped name by issuing the DOS **DIR** command.

Use the mapped names shown in the directory listing whenever you need to refer to UNIX files in a DOS command.

Examples of mapped filenames

The following table illustrates the operation of DOS Services filename-mapping on various types of UNIX filenames. The UNIX name is shown in the left column. A typical mapped name is shown in the right column.

UNIX name	Mapped name
Mail	MAIL'FPE
messagetoall	MESS'BAQ
message.tobob	MESS'BBF.TOB
+.toomuch	_'PS.TOO
:rofix	_ROF'CBL
:rofix.xtn	_ROF'BPQ.XTN
=.ok	_'PP.OK
MiXcAsE.xtn	MIXCA'5U.XTN
okbase.:=+	OKBAS'QW.___
a.b.c	A_B'SV.C

This table illustrates the filename-mapping system:

- Characters that are not legal DOS characters (like " + ", " : ", and " = ") are all mapped to an underscore (_).
- Up to three characters following the rightmost period in the UNIX name are preserved in the mapped name as a DOS filename extension.
- Both lowercase and uppercase letters in the UNIX name are presented as uppercase in the mapped name.
- The index following the apostrophe can consist of either letters or numbers and can be of varied length in different names.

Note that you need to use mapped filenames only when you use files created under the UNIX system with names that are not legal DOS names.

Displaying UNIX-style directory listings

Although you always use a UNIX file's mapped name with DOS commands, you sometimes want to know the original UNIX file- or directory name. The DOS Services **udir** command displays the contents of a UNIX directory in a format that combines the UNIX command **ls -l** and the DOS **DIR** command. The first two fields show both the UNIX name and its corresponding mapped DOS name.

The **udir -a** option displays "hidden" UNIX files. These are UNIX files with names that start with a period, which are normally not displayed in a directory listing.

Using DOS drives

Drive letters are used under DOS Services the same way they are under raw DOS. However, DOS Services imposes certain additional conventions and limitations. Certain drive names should always be associated with certain devices, for instance.

Drive letters *A:* **and** *B:* should be used only with devices that are functionally equivalent to floppy disks. This includes physical floppies and virtual floppies.

Drive letters *C:* **and beyond** are used for devices which DOS Services treats as hard disk drives. These can include:

- The UNIX partition used to hold the hard disk filesystem shared by the DOS and UNIX systems.
- Virtual DOS partitions, which are portions of the shared filesystem set aside to emulate DOS disks.
- Physical DOS partitions on the same disk where the shared filesystem resides.
- Actual DOS-formatted hard disks which are separate from the hard disk used to hold the filesystem shared by the DOS and UNIX systems.

Drives *C:*, *D:* **and** *J:* are used to access the partition shared by DOS and the UNIX system.

Drive *E:* is the default designation for a physical DOS partition on the main disk (the same disk that holds the partition shared by the DOS and UNIX systems). The DOS partition is a special section of the fixed disk reserved for DOS files. If no physical DOS partition is present, *E:* can refer to a virtual DOS partition. See "Using physical DOS partitions" (page 159) for more about physical DOS partitions.

Drive letters *E:* **through** *I:* can be used to refer to either physical or virtual DOS partitions. No letters beyond *I:* should be used for virtual partitions.

DOS Services automatically allocates room for drives up to and including N:. If you need more, you must specify that with the **LASTDRIVE** command in the *CONFIG.SYS* files used in DOS image construction. (See *Administering DOS Services* in the *System Administrator's Guide* for details.)

Drive letters K: through LASTDRIVE can be used for other DOS devices, like CD ROM drives. The drivers for these devices are loaded in the *CONFIG.SYS* file. Starting with K:, DOS Services automatically assigns the next available drive letter to each such device.

Another purpose for high drive letters is for use with the **SUBST** command. **SUBST** is often used to equate a long pathname string to a two-letter drive string.

```
subst m: d:\clients\reports\monthly\june
```

Having a series of assignments like the one above can save you a lot of keystrokes when doing operations that require hopping back and forth between directories.

For most purposes, DOS drives *C:* and *D:* are the most convenient drives to use when you install and run DOS commands and applications.

The following table illustrates these drive letter conventions under DOS Services and the sections which follow provide details.

Drive Letter	Used for
A:	Floppy disk (real or virtual)
B:	Floppy disk (real or virtual)
C:	Accesses the shared filesystem, starting at root.
D:	Accesses the shared filesystem, starting at user's *HOME* directory.
E:	Default designation for the physical DOS partition on the DOS Services fixed disk. Can also be used for virtual DOS partition.
F:	Available for virtual or physical DOS partition.
G:	Available for virtual or physical DOS partition.
H:	Available for virtual or physical DOS partition.
I:	Available for virtual or physical DOS partition.
J:	Accesses the shared UNIX/DOS filesystem starting at *\usr\ldbin.*
K: to LASTDRIVE	Available for other DOS devices.

Drive D:

Your own files and directories on the DOS Services fixed disk are accessible on drive *D:* just as they are on drive *C:*. On drive *D:*, however, your UNIX **$HOME** directory is the *root* of the DOS filesystem. That is, if you are logged in as the user *ELAINE*, the directory *D:\* contains the same files as *C:\USR\ELAINE*. Because your home directory is the root of the filesystem on drive *D:*, you cannot move upward. This means you can only use drive *D:* to access files in your home directory or the subdirectories beneath it. It also means Drive *D:* is unique for each user.

Drive *D:* is useful for installing and running some DOS applications that modify or create files in the root directory. When you install such applications on drive *D:*, they modify or create files in your home directory rather than altering the systemwide root directory (C:\). See *Administering DOS Services* for further information on installing DOS applications.

Drive E:

Drive E: gives you access to the *physical DOS partition,* if available. This is a special section of the fixed disk that is reserved exclusively for DOS work. Drive *E:* is usable only if the system administrator has created and formatted the DOS partition. UNIX files cannot be created on drive *E:* like they can on drives *C:*, *D:*, and *J:*. UNIX does not have direct access to DOS files created on drive *E:*. Although drive *E:* does not share the same files as drives *C:*, *D:*, and *J:*, you use it like a standard DOS disk drive.

Drive *E:* contains no DOS files when you first install DOS Services, but as you use your system, you can add DOS programs, files, and directories to drive *E:*.

Drive *E:* is the same for all users. By default, drive *E:* is a public resource. DOS files and directories created on drive *E:* are not owned by specific users or protected by UNIX file protection mechanisms. This means that all users can create files on drive *E:*, and all users have the power to remove or change any file.

Write access to drive *E:* is available on a first-come, first-served basis. As long as nobody is writing a file on drive *E:*, all DOS Services users can read any file on drive *E:*. When someone writes to a file on drive *E:*, no other users can *read or write* to drive *E:* until the DOS process that is writing on drive *E:* exits.

If your computer has multiple DOS partitions on several fixed disks, DOS Services checks them all. Drive *E:* always accesses the *first* primary partition DOS finds, which is usually on the first disk. For further information on the physical DOS partition, see *Administering DOS Services*.

DOS

Drive J:

Drive *J:*\ contains the same files as *C:\USR\LDBIN*. Because *\USR\LDBIN* is the root of the filesystem on drive *J:*, you can use the *J:* drive to install public applications that must be installed in the root directory.

That is, Drive *J:* is mated to the DOS applications directory, *C:\USR\LDBIN* with the DOS **SUBST** command. When you change to drive *J:* the root of the DOS filesystem is *\USR\LDBIN*.

Virtual DOS floppies and virtual DOS partitions

Your DOS Services system may have one or more virtual floppy drives or virtual DOS partitions. These are files within the shared filesystem that are formatted as DOS volumes. A virtual floppy is a UNIX file which emulates the function of a DOS floppy disk. You can store files there and even boot from it. A virtual DOS partition is a UNIX file which emulates the function of a DOS partition on the hard disk.

These virtual drives are not useful in the UNIX environment, but you can use them with DOS as you would use physical DOS diskette drives or physical DOS partitions. *Administering DOS Services* in the *System Administrator's Guide* describes how you create and administer virtual floppies and partitions.

By default, virtual floppies and partitions are not automatically accessible when you enter the DOS environment. You must use the **dos +a** option to attach any virtual floppies or partitions you want to use during a particular DOS session. See "Attaching devices with dos +a" (page 154) for instructions.

Virtual floppies and partitions have the same access restrictions as the physical DOS partition (drive **E:**). Multiple DOS processes can read the same virtual floppy or partition at the same time, but when a process writes to the virtual device, no other process can read or write to the device until the writing process exits.

You can use the **dosopt** command as described in *Administering DOS Services* in the *System Administrator's Guide* to configure DOS applications or the DOS environment to attach specific virtual floppies or partitions automatically. For further information on attaching virtual floppies and partitions, see the description of the ±a option in "Attaching devices with dos +a" (page 154) or the **dos**(ADM) manual page. In addition, for more information about attaching devices while using the Desktop, see "Using the DOS options dialog box" (page 151).

Reassigning DOS Services drives

Unless you intend to change standard DOS Services functionality, do not use the DOS **ASSIGN**, **JOIN**, or **SUBST** commands to redefine drives *C:*, *D:*, *E:* or *J:* so they refer to other drives or directories. You can, however, use these commands to make other DOS drives refer to the standard DOS Services drives without affecting DOS Services functionality. For example, the following command defines DOS drive M: so that it refers to the subdirectory *REPORTS\MONTHLY* within your home directory (*D:\*):

```
subst m: d:\reports\monthly
```

For more information on these three commands see your DOS documentation or the **assign**(CMD), **join**(CMD), or **subst**(CMD) manual pages.

Chapter 16

Working with DOS files

DOS and the UNIX systems use different file naming conventions. See "Naming DOS files and directories" (page 128) for an explanation of the differences, along with instructions for working with both types of filenames at the same time.

DOS and the UNIX systems also store text files in different formats. The UNIX system stores text lines as a sequence of characters terminated by a line-feed character. DOS, on the other hand, terminates text lines with a carriage-return character followed by a line-feed character. A file created in one format can appear corrupted when accessed by the other.

Converting DOS and UNIX files

When you use DOS on your SCO system, you can use any file that was created with DOS because these files are stored in DOS format even when they are created on the shared UNIX/DOS filesystem. To use a text file in UNIX format with DOS programs, however, you must convert the file to DOS text format using either the DOS Services **unix2dos**, or the UNIX **xtod**(C) command. For example, if you are running DOS Services convert the file *letter* in UNIX format to the file *ltr.dos* in DOS format by typing:

unix2dos *letter ltr.dos*

You can also convert the file and copy it from one drive to another in one step, as the following example illustrates.

unix2dos *c:bdgtmemo a:budget*

When you create text files with DOS that you want to use later with UNIX utilities, you can convert them to UNIX text format with either the DOS Services command **dos2unix** or the UNIX command **dtox**(C). For example:

```
dos2unix memo memo.unx
```

You can use the DOS Services commands, **unix2dos** and **dos2unix**, both in the DOS environment and from the UNIX shell. When you enter the **unix2dos** or **dos2unix** command in the DOS environment, you use DOS filenames, including mapped names when appropriate. When you use these commands from the UNIX shell, use UNIX (unmapped) names. The following example converts the file *message.tobob* (which would have a mapped name in the DOS environment) from UNIX format to DOS format and names the DOS file it creates with a legal DOS name:

```
$ unix2dos message.tobob message.bob
```

You can combine these commands with other DOS or UNIX commands through pipes and redirection. For example, the following command converts the file **names** from DOS format to UNIX format, sorts the text, and appends the sorted text to the UNIX file **newnames**:

```
$ dos2unix names | sort >> newnames
```

Do not specify the same name for the source file and the target file or try to redirect your output back into the source file. The following examples are incorrect:

```
$ dos2unix names names # incorrect
$ dos2unix names > names # incorrect
```

When you omit the target filename, **unix2dos** and **dos2unix** display the text file conversions but do not save them.

When you do not know the format of a text file, you can use the **unix2dos** or **dos2unix** command to convert to the format you need, just to be sure. The commands do not change anything when the file is already in the target format.

NOTE Use **unix2dos**, **dos2unix**, **xtod**(C) and **dtox**(C) only on ASCII text files. These commands do not convert programs, database files, or special-format files created by some word processors.

Accessing other users' files

DOS Services users have limited access to files owned by other users. Whether or not you can inspect or modify other users' files depends on how UNIX permission modes are set on your computer. See "Changing access permissions" (page 80) for UNIX access control. All DOS and UNIX files and directories you create or access in the shared UNIX/DOS filesystem are protected by these permission assignments.

DOS Services, unlike a conventional DOS system, is designed to accommodate multiple users. It therefore provides tools for preventing inspection, alteration, or execution of files by unauthorized users. In general, you cannot modify or delete a file or directory that belongs to someone else.

With DOS Services, you can restrict access to your files so that unauthorized users cannot see the contents of your directories or read your files. On the other hand, you can also grant other users permission to modify or delete your files and directories if you so choose. See "Changing access permissions" (page 80) and "Controlling access to files" (page 30).

The following default permissions are typical:

- Users can inspect the contents of any directory with the **DIR** command.
- Users can read the contents of any file (with the **TYPE** command, for example). Users can also copy any file to their own directories.
- Users can run programs contained in any directory.
- Users cannot modify or delete files or directories belonging to other users.

DOS applications and file permissions

Remember that most DOS applications are designed for a single-user environment. When used with DOS Services in a multiuser environment, most DOS applications do not protect your files from being simultaneously updated by you and another user with write permission.

You should consider carefully which combination of file and directory permissions give you the most appropriate protection. For example, to prevent a file from being simultaneously updated by someone else while you are working on it, you could temporarily remove execute permission for the directory containing that file for all other users. This would prevent anyone from even looking at the file until you were done. Alternatively, you could remove everyone else's write permission for the file. This would allow others to look at a file you are working on, but not to update it. Note that these measures do not protect a file if another user has opened the file and is using it at the time you change permission modes.

Newer DOS programs that use locking calls can prevent these problems without any special user action.

Printing from the DOS window

All standard DOS print functions work in DOS Services. These functions include the print screen (⟨Prt Sc⟩) key, the **PRINT** command, the **COPY** command, and printing operations performed by DOS applications.

By default, DOS Services sends DOS printer output via the UNIX print spooler to a printer named **doslp.** (Your system administrator must set this printer up before it can be used.) The following describes printing procedures you can use with the default DOS Services configuration.

Printing from DOS applications

DOS Services stores printing sent by DOS applications to any of the DOS parallel ports (LPT1, LPT2 or LPT3) in a temporary file. It is printed when either of two conditions occurs:

- You exit the application and return to your DOS prompt, or
- More than 15 seconds have elapsed since the application sent a character to be printed.

Printing with the DOS COPY command

You can print by using the DOS **COPY** command exactly as you would under standard DOS:

copy ***filename*** **prn**
copy ***filename*** **lpt1**
copy ***filename*** **lpt2**
copy ***filename*** **lpt3**

Printing with the DOS PRINT command

To print a file using the DOS **PRINT** command, type the command in the form:

print ***filename***

You cannot use **PRINT** options (such as /T, /C, and /P) when you use the UNIX spooler.

Printing with the Prt Sc key

Press the ⟨Shift⟩ and print screen (⟨Prt Sc⟩) keys at the same time to print the current screen contents just as you would under standard DOS.

To use the ⟨Prt Sc⟩ key to save and print your screen contents as the screen is updated:

1. Press ⟨Ctrl⟩⟨Shift⟩⟨Prt Sc⟩ once to start saving your screen contents. You can then continue to perform operations that change the appearance of your screen. DOS Services saves all changes in a temporary file until you are ready to print.
2. Press ⟨Ctrl⟩⟨Shift⟩⟨Prt Sc⟩ a second time to stop the accumulation of screen contents and start printing.

The "printer not ready" message

Printing from some DOS applications may fail and produce an error message similar to the following:

```
Printer not ready.
```

To print from these applications, attach the printer directly to the DOS process using the following steps:

1. If the printer is currently used to print spooled UNIX print jobs, the system administrator must use the **disable** command to disable the printer.
2. Use the **+a** option to attach the printer you want to use (For more information on the **+a** option, see "Attaching devices with dos +a" (page 154) or the **dos**(ADM) manual page). DOS Services uses the device names *lp0*, *lp1*, and *lp2* to identify the first, second, and third parallel printer ports. Use the name that corresponds to the port to which your printer is attached. For example, if your printer is attached to */dev/lp0*, you can start the DOS environment with the command:

   dos +alp0

   Consult the manuals for your computer and the DOS Services */etc/dosdev* file if you are uncertain how to identify your printer port.
3. You can now start the application and send data to the printer.
4. When you are finished using the printer that is directly attached to DOS, the system administrator can reenable it for UNIX printing by using the **enable** command. UNIX printing cannot be enabled if the DOS process using the directly attached printer is still running.

Chapter 17

Controlling the DOS work environment

When you use SCO Open Desktop/SCO Open Server DOS Services to run DOS commands and applications, you see the same behavior you would see on a conventional personal computer running the same commands and applications. This is possible because DOS Services creates a *virtual personal computer* (also called a *virtual PC* or *virtual machine*) for you whenever you run DOS. A virtual PC has all the important characteristics of a real stand-alone, single-user computer based on the Intel 8086 processor. Instead of existing as physical hardware, however, a virtual PC is a simulation of personal computer hardware. The simulation uses the virtual 8086 mode of your Intel 80386 or 80486 processor and configures resources such as memory so that DOS can use these resources the same way it uses a real personal computer. For example, DOS Services by default allocates 1 Mbyte of memory to your virtual PC. Any software running under the control of the virtual PC can use this memory the way it would use the memory on a stand-alone 8086 computer.

SCO Open Desktop/SCO Open Server can create more than one virtual PC at a time, which allows users to run several DOS tasks at once. Each DOS environment under DOS Services runs in its own separate, protected, virtual machine, which cannot harm the operation of other DOS environments or the UNIX system. In particular:

- DOS programs cannot disable system interrupts. They can only disable their own "virtual" interrupts, which affect only that one DOS environment and not any other DOS environment or the UNIX environment.
- Errant DOS processes cannot damage UNIX processes or other DOS processes because each DOS environment is assigned a specific segment of memory and cannot write outside it.
- DOS programs can only affect I/O devices that are assigned to them, and not those assigned to the UNIX system or other DOS programs.

You can customize a virtual PC in much the same way you can customize a conventional stand-alone personal computer. For example, if you run DOS applications that need more than the default memory, you can add expanded memory. If you use a DOS application that needs a COM port, you can add one. When you use DOS Services, however, you do not open your computer and install adapter cards containing memory chips or a COM port. Instead, you use simple command options that tell DOS Services to configure these resources—which are already physically present—so they become part of your virtual personal computer. Because each virtual PC is independent of all others, you can customize each one as appropriate for the applications running in it.

DOS Services uses *DOS images* to improve efficiency. A DOS image is a frozen picture, or snapshot, of DOS after it has been loaded into memory and is running. This image includes information DOS needs about the virtual PC configuration. When you start DOS from your UNIX shell or from the Desktop, a virtual PC is created and a DOS image is loaded into that virtual PC's memory. This procedure has the same effect as booting DOS on a conventional personal computer, but it is much quicker.

Controlling the DOS window

You can control the the DOS window with the **DOS** menu.

There are two ways to invoke the **DOS** menu. To invoke the **DOS** menu with the mouse, move the cursor into the DOS window and press and hold the right mouse button. The **DOS** menu appears. Keep holding the button down and move the mouse cursor to the desired option on the menu, and then release the button. The option is selected and the **DOS** menu goes away.

You can also invoke the **DOS** menu by pressing a special key sequence. By default it is ⟨Alt⟩**D**, but it can be redefined. The key sequence works only when the DOS window is already selected. To select one of the options on the menu, click on it with either the left or right mouse button. To close the **DOS** menu without selecting any option, simply click the mouse outside of the **DOS** menu.

The options on the **DOS** menu are as follows:

Zoom	Zooming a window means causing it to expand so it fills your whole screen. When you want to run a DOS EGA/VGA graphics program, you must zoom the window it is in; you cannot run such a program in a normal window. You can also zoom any other window if you want it to take over the screen. To unzoom (that is, return a zoomed window to its

	default size), press the DOS menu key sequence (⟨Alt⟩D by default).
Focus [Unfocus]	Selecting the Focus option allows your mouse input to be passed to any DOS program that is running in the DOS window and capable of using mouse input. During this time the mouse is devoted to DOS and cannot be used to move the DOS window or for any other X pointer task. Selecting the Unfocus option returns the mouse to X pointer status and allows you to use the mouse in other X windows. Only one of these options (Focus or Unfocus) appears on the DOS Menu at a time. When the mouse is already focused in the DOS window, it cannot be used to bring up the DOS Menu. You must use the DOS Menu key sequence.
Refresh	Selecting the **Refresh** option redraws the DOS window.
DOS Colors [X Colors]	Selecting the DOS **Colors** option sets the colors for your DOS window to the sixteen standard text colors. Select the **X Colors** option sets it back to the colors chosen through the **Color** utility. Only one of these options appears on the menu at a time.
Autofreeze	Selecting the Autofreeze option cause your DOS session to be suspended as soon as you move your X pointer outside the DOS window. "Freezing" saves processor cycles for other work. You may also find it useful when you are monitoring activity in the DOS window and do not wish to miss any changes. You can set a DOS process in motion and then do other work on the desktop. Your DOS process resumes its activity only when you once again move the X pointer inside the DOS window.
Quit	Selecting the **Quit** option closes your DOS session.

NOTE **Autofreeze** and **Focus** are mutually exclusive. The **Autofreeze** option is used only when your mouse is functioning as the X windows pointing device. It cannot be used when **Focus** is on. The **Focus** option controls shifting the mouse back and forth from X pointer status to DOS pointer status.

Some applications run in graphics mode only part of the time. When using such an application, you only need to zoom it when it enters graphics mode. For example, a spreadsheet can work as a text application but it can also draw graphs. In a case like this, you could perform text entry in a regular DOS window without worrying about zooming. If your application enters graphics mode, however, a message displays reminding you to zoom, and the DOS window running your program becomes unusable until you have done so. You can return to the normal DOS window after you are finished using graphics mode by pressing the **DOS** menu key sequence (usually ⟨Alt⟩**D**).

Note that you can unzoom at any time, even if you are still in EGA/VGA graphics mode, in order to use other clients. However, when you unzoom while your application is in EGA/VGA graphics mode, the application in the window is suspended until you zoom it again.

When the DOS program you are running requires a mouse, use the **DOS** menu key sequence; then select the Focus option from the **DOS** menu to focus your mouse in the DOS window. This means that your mouse input goes to the DOS program instead of to the server. The **Unfocus** option reverses this, so that your mouse input goes to the server. If you select the **Zoom** option, your mouse is automatically focused for you while you are zoomed.

Changing colors

By default, DOS Services uses the current UNIX color palette to display the colors in the DOS window. For many DOS programs this is sufficient. For DOS programs that depend heavily on color, the results can be peculiar. When the colors DOS Services expects to use are not available, you may find that objects in their DOS windows are displayed in unexpected colors.

The colors you see in your DOS program depend upon a number of factors: your hardware setup, the colors chosen for your X Windows session, and the color requirements of the DOS program you are running. DOS programs run in X windows may produce distorted colors or unreadable screens when run on 16-color servers.

The **DOS Colors** option on the **DOS** menu allows you to have your DOS windows displayed in true DOS colors. This can have an unexpected impact on the appearance of your UNIX windows. Alternatively, you could use the Desktop **Color** control (page 58) to select a special DOS palette. Some DOS applications let you select colors specifically for that application.

See *Administering DOS Services* in the *System Administrator's Guide* for more about controlling colors in the DOS window.

Using AUTOEXEC.BAT and CONFIG.SYS files

DOS interprets the commands in two special files automatically every time you enter the DOS environment. These files are *AUTOEXEC.BAT* and *CONFIG.SYS*.

You can use *AUTOEXEC.BAT* to customize your DOS environment or to run commands you want executed every time you use DOS. For example, if you run a program called **GRAPHS** every time you use DOS, you could include the command to run the program in your *AUTOEXEC.BAT* file.

The *CONFIG.SYS* file (if it exists) contains information about your computer's configuration that the system needs to know every time you run DOS. Some DOS applications, for example, require device drivers that are identified in *CONFIG.SYS*.

Because different users may want to include different commands in their *AUTOEXEC.BAT* or *CONFIG.SYS* files, DOS Services provides for both:

- System default *AUTOEXEC.BAT* and *CONFIG.SYS* files, which affect all users unless they explicitly specify otherwise.
- Personal *AUTOEXEC.BAT* and *CONFIG.SYS* files, which individual users can create to customize their own personal DOS environments.

If you create a personal *AUTOEXEC.BAT* file in your home directory, DOS Services executes it whenever you enter the DOS environment or start a DOS process. DOS Services executes your home directory *AUTOEXEC.BAT* file after executing the root directory *AUTOEXEC.BAT*.

In general, DOS Services interprets *CONFIG.SYS* commands just as conventional DOS personal computer does. However, there are some exceptions. Notable exceptions are as follows:

BUFFERS DOS Services does *not* interpret **BUFFERS** commands in any of your system's *CONFIG.SYS* files at DOS run time. The **BUFFERS** value is defined in the DOS images at the time they are created and cannot be changed unless you make new DOS images. The **BUFFERS** value used in the factory default DOS images is 15, the standard DOS default value for 640K of RAM. See *Administering DOS Services* for further information on changing **BUFFERS** and making new DOS images. The **BUFFERS** value is effective only when you use an actual DOS file system. It is not used when you access the shared DOS/UNIX filesystem.

COUNTRY	It works just as it does on a conventional DOS personal computer, except that the third parameter (drive and pathname of your country information file) is not supported. DOS Services always looks for the country information file in the root of your current drive.
DOS	determines the area of memory into which DOS will be loaded and specifies whether the upper memory area can be used for drivers and TSRs. The DOS command is already defined in your DOS images at the time they are created. The value used in the factory default DOS images is DOS=HIGH,UMB, which loads DOS into the High Memory Area and allows the Upper Memory Blocks to be used. This provides the maximum RAM for DOS programs. It should not be changed. See *Administering DOS Services* for further information on making new DOS images.
DRIVPARM	sets the characteristics of a disk drive. It works just as it does on a conventional DOS personal computer. It is effective only when you access actual DOS filesystems.
FCBS	sets the number of File Control Blocks that DOS can open concurrently. DOS Services does *not* interpret **FCBS** commands in any of your system's *CONFIG.SYS* files at DOS run time. The **FCBS** value is defined in the DOS images at the time they are created and cannot be changed unless you make new DOS images. The **FCBS** value used in the factory default DOS images is the same as the standard DOS default value: 4. See *Administering DOS Services* for further information on changing **FCBS** and making new DOS images. The **FCBS** command is effective only when you use an actual DOS file system. It is not used on any portion of the shared DOS/UNIX file system.
FILES	The **FILES** command sets the maximum number of files that DOS can open concurrently. If there is more than one **FILES** command in the *CONFIG.SYS* files that DOS Services interprets when DOS starts, the *highest* value is used. The **FILES** command is effective only when you use an actual DOS file system. The limit for a UNIX file system is set separately and is not affected by the **FILES** command. A DOS process can open a maximum of 123 files simultaneously in the shared DOS/UNIX file system.
STACKS	sets the amount of RAM to be set aside for the processing of hardware interrupts. DOS Services does *not* interpret the **STACKS** variables in any of your system's *CONFIG.SYS* files at DOS run time. The **STACKS** value is defined in the DOS images at the time they are created and cannot be changed unless you make new DOS images. The **STACKS** value used in the factory

default DOS images is the same as the standard DOS default value: 9,128. That is, there are nine stacks with 128 bytes each. See *Administering DOS Services* for further information on changing **STACKS** and making new DOS images.

SWITCHES specifies the use of a conventional keyboard even though and enhanced keyboard is installed. DOS Services does *not* interpret the **SWITCHES** values in any of your system's *CONFIG.SYS* files at DOS run time. The **SWITCHES** value must be defined in your DOS images. The factory default DOS images do not contain a **SWITCHES** value. This means that your enhanced keyboard will function in its enhanced mode.

The **SWITCHES** value cannot be changed unless you make new DOS images. If you are using an enhanced keyboard and you wish it to behave as a conventional keyboard, you must make a new DOS image. See *Administering DOS Services* for further information on making new DOS images.

Configuring memory

DOS Services provides you with a Virtual Personal Computer which emulates the functions of a standard 8086 processor in nearly every way. The one significant exception is added functionality to support Windows 3.1 Standard Mode (80286).

DOS 5.0 provides tools for maximizing the amount of Conventional DOS Memory available for your programs. These tools cannot be used on an 8086 computer, but you can use them under DOS Services. You can load device drivers and TSRs (Terminate and Stay Resident programs) into the Upper Memory Blocks (UMBs) and you can load DOS itself into the High Memory Area (HMA).

Most of this has already been done for you. By default, when DOS Services comes up it loads a special extended memory manager, *MRGXMS.SYS*, which provides access to the Upper Memory Blocks. It also loads DOS into the High Memory Area.

The commands for doing this are built into the DOS images and cannot be changed without making new images. Since this provides an optimum amount of RAM for your programs, you will probably never need to change it. With DOS and all device drivers and TSRs loaded high, DOS Services provides about 610K of Conventional DOS Memory for programs.

The following commands are built into the DOS images:

```
device=mrgxms.sys
dos=high,umb
```

MRGXMS.SYS is the only Extended Memory Manager that can be used with DOS Services. Do not use *HIMEM.SYS*, *XMS.SYS*, or any other DOS extended memory manager produced by third-party vendors.

As in standard DOS, once the Upper Memory Blocks have been enabled, you can free up DOS Conventional Memory by loading your device drivers and TSRs there. Refer to your DOS manuals for more about DOS memory management.

To load device drivers and TSRs in the Upper Memory Blocks, use commands such as the following in any *CONFIG.SYS* file which will be interpreted when DOS Services starts:

```
DEVICEHIGH=DEVICE1.SYS
DEVICEHIGH=DEVICE2.SYS
INSTALL=TSR1.EXE
INSTALL=TSR2.EXE
```

To load TSRs into the Upper Memory Blocks, you can also use commands such as the following in any *AUTOEXEC.BAT* file which will be interpreted when DOS Services starts:

```
LOADHIGH TSR1.EXE
LOADHIGH TSR2.EXE
```

Using expanded memory (EMS)

DOS Services supports the Lotus/Intel/Microsoft Expanded Memory Specification (EMS), so you can run any DOS applications that use expanded memory and conform to this specification.[1] DOS Services expanded memory is available in the following sizes: 512 Kbytes, and 1, 2, 3, 4, 5, 6, and 8 Mbytes. The default amount of expanded memory is one Mbyte, but you can easily request any of the allowable values. (You should not request more memory than you need, however, because it wastes system resources.)

Your computer does not need to have actual physical memory in the amount you request when you use expanded memory, and you do not need an EMS memory card to use expanded memory with DOS Services. DOS Services simulates expanded memory by using standard UNIX system virtual memory.

1. *The Lotus/Intel/Microsoft Expanded Memory Specification*, Version 4.0, Lotus Development Corporation, Intel Corporation, and Microsoft Corporation.

Provided you have at least the minimum amount of memory required to run DOS Services, you can use any of the expanded memory values that DOS Services supports.

To use expanded memory, you must request that a certain amount of it be allocated to a particular DOS process. You can do that through the DOS Options Dialog Box or with the **dos +a** command.

From the Desktop, bring up the DOS Options Dialog Box with the mouse technique described at the beginning of this appendix. Then click on the button marked EMS. A drop down menu shows you increments of EMS memory you can assign. Click on the desired amount and then click the Start button.

To achieve the same effect from the command line, start your process through the **dos** command with the **+a** option. To request the standard one Mbyte of expanded memory, use the **+aems** option from the UNIX command line. For example:

```
dos +aems
dos +aems 123
```

The first example starts a DOS environment with one Mbyte of expanded memory. The second example starts Lotus 1-2-3 with one Mbyte of expanded memory.

To request a different amount of expanded memory, use one of the following **+a** options:

DOS Option	Memory
+aems512	512 Kbytes
+aems	1 Mbyte
+aems1	1 Mbyte
+aems2	2 Mbytes
+aems3	3 Mbytes
+aems4	4 Mbytes
+aems5	5 Mbytes
+aems6	6 Mbytes
+aems8	8 Mbytes

For example, to request four Mbytes, type:

```
dos +aems4
```

Note that **+aems1** has the same effect as **+aems**.

When Expanded Memory was introduced, it was available only as an adapter board. The card was a separate device which had to be attached to the computer like other peripherals. Even though DOS Services emulates EMS in a completely different manner, the process by which you make it available to your program reflects its origin. EMS memory is "attached" with the +a option. Compare this with the method used to grant access to Extended Memory, described in "Using extended memory (XMS)".

Using extended memory (XMS)

Extended Memory is made available to your programs by procedures that are similar but have some essential differences.

Extended Memory is not treated as hardware, so you do not use the "attach devices to DOS" option. It is described in this appendix merely for your convenience, so that you can see clearly the differences and similarities between adding Expanded Memory and Extended Memory.

The **dos** command also offers a **+m** option, which means "set *m*emory size." You can set memory size smaller for those DOS programs which require less than the standard 640 K, or you can allocate memory beyond 1 Mbyte to Windows 3.1.

Note that this additional memory can be utilized *only* by Windows 3.1. Windows can then allocate this memory to its applications.

When you provide memory in this way, Windows 3.1 manages the memory for you. You do not need to be concerned about whether it is "conventional" or "upper memory block" or "extended" memory. For this reason it is referred to simply as Standard memory.

To allocate Standard memory to Windows 3.1 from the Desktop, bring up the DOS Options Dialog Box with the mouse technique described in "Using the DOS options dialog box" (page 151). Move down to the Memory box and click on the button marked Standard. A drop down menu shows you increments of memory you can assign. Click on the desired amount and then click the Start button.

To achieve the same effect from the UNIX command line, start your process through the **dos** command with the **+m** option. The command:

```
dos +m4 win
```

requests four Mbytes and runs Windows.

For more on the +m option, see the *Administering DOS Services* section of the *System Administrator's Guide*, and the **dos**(ADM) or **win**(ADM) manual pages.

Using peripheral hardware with DOS Services

Because DOS Services is a fully configurable environment, you may run many different kinds of DOS sessions. You may configure DOS Services to run in VGA mode with one program and in CGA mode for another. You might run one program with minimal memory to conserve resources and allocate 5 megabytes of EMS memory to another. You might attach a local dot-matrix printer when you use your database program, while you attach a network laser printer for desktop publishing.

When you use many different DOS programs you usually run many different DOS environments. Many users set things up so that the devices they need are automatically attached whenever a particular DOS program is run. Since this device attachment takes place automatically and invisibly, it is common to forget exactly which devices are available at the moment.

Using the DOS options dialog box

From your Desktop, you can access the DOS Options Dialog Box. This is a menu you can use to start a DOS environment or a particular DOS program with a specific hardware arrangement. The DOS Options Dialog Box is available for your DOS sessions, your Windows sessions, or any DOS program which appears as an icon on your Desktop.

The DOS Options Dialog Box allows you to configure your DOS processes with most of the devices you will need. You have your choice of video standards (VGA, CGA, MDA or Hercules). You can associate any available DOS partition with any of the listed drive letters (*E:*, *F:*, *G:*, *H:*, and *I:*). You can associate your DOS printer ports with the UNIX spooler or with direct attached parallel printers.

You can also allocate extra memory to your DOS process. Memory can be added in the form of Expanded Memory (EMS) or "Standard" memory. Standard Memory means the sum of 1 Mbyte of conventional DOS memory plus as many additional Mbytes of Extended Memory you care to assign. Standard memory is limited to 15 Mbytes total.

To access the DOS Options menu, position your mouse pointer over the program's icon and double click with your middle mouse button. If you have a two button mouse, double click with both buttons simultaneously.

When the DOS Options Dialog Box appears, click the appropriate buttons to attach the hardware you want. Then click Start. Your DOS session or program starts with the particular hardware setup you have specified. When you exit that session or program, those devices are detached.

It is also possible to attach devices that are not actually available. If you do, restart your process with appropriate hardware. For instance, suppose you attach COM1 to your DOS session and that port is already in use:

1. When your process starts you see a message like this:

```
INTFDEV_FAIL:DOS:error:could not initialize '/dev/vcom1/dev/tty1a.'
```

2. Acknowledge by clicking on OK. This is followed by a second error message:

```
VM_DIED:DOS:ERROR: The VM86 process died.
```

3. Once again, acknowledge by clicking OK.
4. Bring up the DOS Options Dialog Box again by double clicking your DOS icon. Use the center button on a three-button mouse or both buttons on a two-button mouse.
5. Select another COM port and click on Start.

Using the apply button

The DOS Options Dialog Box also offers a button marked Apply. This allows you to save your hardware choices and use them for all future DOS sessions or all future invocations of that particular program. Use the Apply button with restraint. Remember that any memory or hardware you attach is temporarily unavailable to other users or other programs you might run. Use Apply to save only the minimum configuration you need to run DOS or Windows.

This is especially important for COM ports. Any time you attach a COM port to your session it is unavailable to other users. Attach COM ports only when you actually plan to use them.

In addition to the DOS Options Dialog Box the entire range of devices can be accessed through the **dos** +a or **win** +a commands. For more information on attaching devices with either command, see "Attaching devices with dos +a" (page 154), or the **win**(ADM) and **dos**(ADM) manual pages. Advanced users should also see the *Administering DOS Services* section of the *System Administrator's Guide* for more information on the attaching and configuring devices to DOS.

Using the device information window

The Device Information window gives you instant access to this information any time you are in DOS. It works from the DOS command line or from within any DOS application. It "pops up" when you hit a hot-key sequence, like a TSR program in the standard DOS environment.

To open the Device Information window, press ⟨Ctrl⟩⟨Esc⟩, then ⟨Ctrl⟩⟨I⟩. While the Device Information window is displayed, your DOS process is suspended. Any other processes, (DOS or UNIX) which may be running in the background continue to run. When you have obtained the information you need, press ⟨Space⟩ and you pop back into your DOS session, which picks up exactly where you left it.

The Device Information window is only available when your DOS window is in text mode.

The default color scheme depends on your system setup. It is usually white on black or yellow on red. If you find this hard to read or prefer another arrangement, it is configurable.

You can reset the colors of the Popup Window by setting the UNIX environment variable, **DOSCONFIG**. The following two commands show how this is done from the Bourne shell:

```
DOSCONFIG=menucolor.white.blue
export DOSCONFIG
```

The **menucolor** option allows you to specify two parameters, separated by dots. The first parameter is the foreground color; the second is the background color. The code above gives you white characters on a blue field. Sixteen colors are available for each. See *Administering DOS Services* in the *System Administrator's Guide* for a list. It is usually white on black or yellow on red. If you find this hard to read or prefer another arrangement, it is configurable.

You can reset the colors of the Popup Window by setting the UNIX environment variable, **DOSCONFIG**. The following two commands show how this is done from the Bourne shell:

```
DOSCONFIG=menucolor.white.blue
export DOSCONFIG
```

The **menucolor** option allows you to specify two parameters, separated by dots. The first parameter is the foreground color; the second is the background color. The code above gives you white characters on a blue field. Sixteen colors are available for each.

Attaching devices with dos +a

If you want to use a hardware device that is not automatically available when you use DOS, you request access to it using the **dos +a** ("attach device") option in the form:

dos +a*device_name* [***command***]

The command form **dos +a*device_name*** starts a DOS environment and attaches the requested device to the DOS process so you can use it for the duration of the DOS environment. The command form **dos +a*device_name* *command*** attaches the specified device to the DOS process and also runs the specified DOS command. You can then use the specified device for the duration of the program you start with ***command***. If your specified device is not available (typically because another UNIX or DOS process is using it), DOS Services displays a message informing you that you cannot use the device.

To attach more than one device to a DOS process, use more than one **+a** option. For example:

dos +acom1 +aems

Examples illustrating the use of the **+a** option appear throughout this chapter. See *Administering DOS Services* in *System Administrator's Guide* or the **dos**(ADM) manual page for descriptions of other useful procedures, including the use of the **dosopt** command to configure DOS commands so they automatically request required devices.

Using display adapters and serial terminals

DOS Services automatically senses the type of display adapter you use in the system console and properly displays DOS processes. DOS Services is compatible with VGA, CGA, Hercules, and monochrome display adapters. When you use a serial terminal, DOS Services displays DOS processes as though they are running on a monochrome console. You can use the DOS Options Dialog Box or the **dos +a** command to explicitly specify a particular display type, but this is normally unnecessary. For further information on display devices, refer to the *Administering DOS Services* section of the *System Administrator's Guide*.

Using a mouse

Note that you should not modify any *CONFIG.SYS* files to identify a mouse driver as you would on a conventional personal computer running standard DOS. DOS Services uses a special mouse driver that is identified in the system default *CONFIG.SYS* file.

DOS Services causes DOS to view any properly configured mouse as though it is a Microsoft Bus Mouse. If you install DOS applications that need to know about the specific mouse you use, always refer to it as a Microsoft Bus Mouse.

If you have a bootable DOS partition on the fixed disk and you sometimes boot "raw" DOS from it, you may have to switch from one mouse driver to another. For more information on switching mouse drivers, see "Switching mouse drivers" (page 197) for more information.

Using a modem

You can use either an external modem (one attached to a serial port) or an internal modem (one that requires an internally installed card) with DOS Services. If you have a choice, consider that external modems are easier to troubleshoot should problems arise.

For either kind of modem, install the modem by following the manufacturer's instructions to connect it to a serial port. Note that an internal modem generally replaces COM1 or COM2. To use the modem, attach the appropriate COM port to your DOS process by using the +a option when you start DOS. For example:

```
dos +acom1
```

For further information, see "Using COM ports".

Using COM ports

DOS can use the COM1 and COM2 serial ports (equivalent to the UNIX devices **/dev/tty1a** and **/dev/tty2a**). The COM3 and COM4 ports are not supported. Only one DOS process at a time can use each COM port. To use a COM port, you must explicitly request access to it. You can use the DOS Options Dialog Box from the Desktop, or the **dos** command with the +a option from the command line.

DOS Services can attach COM ports in two different ways: indirectly or directly. You do not need to understand the technical distinction between these two forms. However, you must choose one form or the other when you start DOS. Consider these trade-offs as you make your choice:

- Indirect attachment is more reliable when the system is heavily loaded, but it may be slower than direct attachment. Try this form of attachment first if you are uncertain which to use.
- Direct attachment is faster but less reliable than indirect attachment when the system is heavily loaded.

Indirect attachment

To indirectly attach a COM port from the Desktop, bring up the DOS Options Dialog Box with the mouse technique described in "Using the DOS options dialog box" (page 151). Click the button beside the COM port you want. Then click Start.

To indirectly attach a COM port from the UNIX command line, use the **dos** command with the **+acom1** or **+acom2** option. For example:

```
dos +acom1
dos +acom2 xtalk
```

The first example starts a DOS environment and requests access to COM1. The second example starts the **CROSSTALK** application and requests access to COM2. In both examples, if the requested COM port is not available, DOS Services does not start DOS and instead displays an error message.

Direct attachment

To directly attach COM1 or COM2, use the **+adcom1** or **+adcom2** option. For example, to start a DOS environment and directly attach COM1, type:

```
dos +adcom1
```

To start **CROSSTALK** and directly attach COM2, type:

```
dos +adcom2 xtalk
```

Direct attachment is not available through the DOS Options Dialog Box.

Using COM ports to transfer files

You can use both directly and indirectly attached COM ports to transfer files between computers. However, the reliability of the transfer depends on many factors including line quality, transfer speed, and system load. If you use COM ports to transfer files at speeds greater than 4800 baud, use an error-correcting protocol to perform the transfer. Error-correcting protocols help ensure the integrity of data during transfer.

Using the game port

To use the game port, use the **+agame** option. For example:

```
dos +agame
```

Only one DOS process at a time can use the game port.

For further information on installing and configuring hardware devices, refer to *Administering DOS Services* in the *System Administrator's Guide* or consult your system administrator or DOS Services distributor.

Using CD ROM devices

CD ROM devices are supported as mounted file systems only. They must be attached through the SCSI bus and the file tree they must be mounted with the **lower** option, which converts all filenames to lower case. This allows DOS to map the filenames correctly. One suitable mount command is shown in the example below:

```
/etc/mount -r -f HS,lower /dev/cdrom /mnt
```

Another option that might be useful is **showhidden**, which makes hidden files show up in directory listings.

Using custom devices

You can install and use hardware devices other than those described in the preceding sections. Your system administrator can define an easy-to-type device name for any device that exists on your computer. You can use any defined device name with the **+a** option. For example, if your system administrator has defined the device name **widget**, you can use the device with this name by typing the command **dos +awidget**.

For further information on installing and configuring hardware devices, refer to *Administering DOS Services* or consult your system administrator or DOS Services distributor.

Using virtual partitions and virtual floppy disks

Virtual DOS partitions and virtual floppy disks are UNIX files that contain actual DOS filesystems. By default, these virtual devices do not exist.

Virtual partitions and floppy disks are typically not of interest to most DOS Services users. If your computer does not have a physical DOS partition, however, you may find a virtual partition to be useful. Refer to *Administering DOS Services* in the *System Administrator's Guide* for further information on creating these virtual devices and on their characteristics.

To use a virtual partition or floppy disk, attach it to your DOS process using the **+a** option in the form:

+a*drive_letter*:=*unix_file_name*

You should normally use drive letters **a** or **b** to access a virtual floppy. Use drive letters **e**, **f**, **g**, **h** or **i** to access virtual partitions.[2] The full pathname of the UNIX file that contains the virtual partition or floppy drive is ***unix_file_name***. For example, the following command starts a DOS environment and attaches the virtual partition */usr/fred/vdisk* as DOS drive F:

```
dos +af:=/usr/fred/vdisk
```

You can then access any DOS files contained within */usr/fred/vdisk* via DOS drive F:. You can change your current drive to drive F: with the command:

```
f:
```

You can list the files on drive F: with the command:

```
dir f:
```

To start a DOS environment and attach a virtual floppy named **/usr/phyllis/vflop**, use a command such as:

```
dos +ab:=/usr/phyllis/vflop
```

When you issue this command, the virtual floppy is accessible as DOS drive B:, just as if it were a physical diskette drive. Note that if there is a physical drive B:, you will no longer have access to it.

Virtual partitions and floppy drives have the following limitations:

- While multiple users can simultaneously attach and read a virtual partition or floppy drive, only one user at a time can write to a virtual partition or floppy drive.
- When one user is writing to a virtual partition or floppy drive, all other users are prevented from reading and writing to that device until the DOS session on the device is terminated.

See *Administering DOS Services* in the *System Administrator's Guide* for further information on attaching virtual partitions and floppy disks, including instructions on attaching them "exclusive" so they cannot be written by other users.

2. You may want to avoid using drive *E:* since it is assigned to the primary DOS partition by default. However, you can use drive *E:* if you wish. Note that if you assign drive *E:* to a virtual floppy or virtual partition, you will no longer have an attachment to the primary DOS partition (unless you specifically assign another letter to it).

Using physical DOS partitions

A physical DOS partition is a portion of the fixed disk formatted as a DOS filesystem and reserved exclusively for DOS files. DOS Services may have just one physical DOS partition (called the primary DOS partition); it may have both a primary and an extended DOS partition; or it may have no physical DOS partitions.

If your SCO system has a primary DOS partition, it is automatically available as DOS drive *E:* whenever you run DOS. If your computer has an extended DOS partition, you must attach each logical drive you want to use to an available DOS Services drive letter.

Be careful not to attach any logical drive twice. It is easy to do this unintentionally with DOS Options Dialog Box. If you do inadvertently assign two different drive letters to the same segment of the fixed disk, DOS Services responds by protecting your files from corruption. All files on the partition are made read-only.

To attach a logical drive within the extended DOS partition, you can use either the DOS Options Dialog Box or the **dos +a** command.

The DOS Options Dialog Box displays a column of drive letters with buttons beside them. To assign a drive letter to a partition, click on the button beside that letter. A drop down list appears. Double click the name of the selected partition from the list. Then click Start to start your DOS program.

From the UNIX command line, use the **dos** command with the **+a** option in the form:

dos +a*DOS Services_drive_letter***:=dos***logical_drive_letter*

DOS Services_drive_letter can be **e**, **f**, **g**, **h** or **i**. *logical_drive_letter* can be any of the logical drives available under raw DOS.

NOTE You may want to avoid using drive *E:* since it is assigned to the primary DOS partition by default. However, you can use drive *E:* if you wish. If you assign drive *E:* to a logical drive in the extended partition, you will no longer have an attachment to the primary DOS partition unless you specifically assign another letter to it.

For example, the following command attaches your system's logical DOS drive *D:* to the DOS Services *F:* drive:

dos +af:=dosd

Refer to the file */etc/dosdev* for a list of available logical drives.

Multiple DOS processes can *read* files on a DOS partition at the same time, but only one process at a time can *write* to the primary DOS partition or to a logical drive within the extended DOS partition. As soon as one process writes to a DOS partition, no other process can read or write to the partition until the DOS session writing the first process is terminated.

Refer to the *Administering DOS Services* section of the *System Administrator's Guide* for further information on using and administering physical DOS partitions.

Chapter 18

Using DOS with the UNIX system

With DOS Services, you can also perform tasks that would be impossible using either a UNIX or DOS system separately. For example, you can create a UNIX shell script that, with a single command, loads a DOS word processor like WordPerfect® so you can create a report, and then sends that report to other users via the UNIX **mail** utility.

The examples in this chapter assume you are using the standard UNIX Bourne shell. If you use a shell other than the Bourne shell, you may have to modify some of the examples.

Running DOS programs from the UNIX system

To run an off-the-shelf application that is installed on your fixed disk, start it from your UNIX prompt just as you would at a DOS prompt.

Most standard DOS commands have already been set up so that they are executable from the UNIX prompt. However, you will have to perform this setup yourself on any DOS application you install. See *Administering DOS Services* in the *System Administrator's Guide* for instructions.

When you run a program in DOS Services, the system must be able to find the program you specify. This rule applies whether you use the DOS environment or the UNIX shell, and whether the program you specify is a DOS or a UNIX program.

Your default UNIX search path includes */usr/dbin* and */usr/ldbin*, the directories where standard DOS commands and applications are stored. You can therefore run all standard DOS commands and applications while working in any directory without typing their full paths on your command line or using the DOS **PATH** command.

If you are familiar with DOS search rules, you should be aware that UNIX and DOS rules for search paths differ in one respect: DOS always looks for any program you execute in your current working directory first, before it checks the rest of your search path. UNIX, on the other hand, does not look in the current directory unless it is explicitly specified in the search path.

Using DOS commands

All common DOS commands work as you expect when you use the UNIX shell. Use these commands just as you would use them in the DOS environment. For example:

```
dir
fc memo memo.old
```

If you choose, you can intermix DOS and UNIX commands:

ls	The UNIX command for a directory listing
del report	The DOS command for deleting a file
cp plan newplan	The UNIX command for copying a file
comp plan newplan	The DOS command for comparing files

(For a list of similar DOS and UNIX commands, see Table 7-1 (page 63).)

At the UNIX prompt, pipes and redirection work as they do on a conventional DOS system. For example:

```
$ dir | sort > contents
```

When you run DOS programs from the UNIX shell, you must use UNIX syntax. For example, use the slash (/) in path names (*/usr/dbin*, for example) and the hyphen (-) to indicate command options (**dir -w**, for example). If you are a UNIX user, this syntax should be familiar to you. If you are accustomed to DOS, however, you need to know when the UNIX shell interprets your commands differently from standard DOS. When you use DOS from the UNIX shell, observe the following rules:

1. Use the UNIX path separator " / " and switch character " - " instead of the standard DOS path separator " \ " and switch character " / ". For example, in the DOS environment you would type:

   ```
   dir \usr\dbin /w
   ```

   At the UNIX shell you would instead type:

   ```
   dir /usr/dbin -w
   ```

2. When working from the UNIX shell, use lowercase letters for DOS commands and filenames just as you do for most UNIX commands and filenames. For example, type:

   ```
   dir /usr/dbin
   ```

Do not type:

DIR /USR/DBIN *(this is incorrect)*

3. When you use the UNIX shell, the following characters are interpreted according to the rules of UNIX syntax, rather than the rules of DOS syntax:

   < > * ? | & $; \ " ` ' ^ ( ) [ ] #

You can therefore use them as necessary with DOS commands just as you would use them with UNIX commands. If you are used to using any of these metacharacters on a conventional DOS system (or in the DOS Services environment), however, you should be aware that the UNIX system treats them differently than DOS does. For example, if you issue the DOS command:

```
copy *.com a:
```

in the DOS environment, DOS copies all files ending with **.com** in your current directory on drive *C:* to your current directory on drive *A:*. The equivalent command from the UNIX shell is:[1]

```
dos copy *.com a:
```

This command works differently than it does in the DOS environment because the UNIX shell translates the asterisk (*) before the DOS **COPY** program receives the command. By the time the command is passed to DOS, it has been translated into something like the following:

```
dos copy file1.com file2.com file3.com a:
```

An error message results since DOS **COPY** accepts only one source filename. Similar errors can occur with other UNIX shell metacharacters.

You can prevent the UNIX shell from interpreting shell metacharacters by using the standard UNIX shell escape character, the backslash (\). For example:

```
dos copy \*.com a:
```

The backslash prevents the UNIX shell from interpreting the metacharacter " * ", and the command is passed to DOS and interpreted correctly.

You can also use both the single quote (') and double quote (") symbols to prevent the UNIX shell from interpreting metacharacters. The quotes in the command:

```
dos copy "*.com" a:
```

tell the UNIX shell not to interpret the metacharacter asterisk (*), and the command is passed to DOS in the desired form. Single quotes work equally well.

1. Note that the command must start with **dos** to avoid a conflict with the UNIX **copy** command.

Some DOS commands require the quote symbol as part of their command syntax. The quote must be passed literally to DOS rather than being used by the UNIX shell to prevent interpretation of other metacharacters. The DOS **FIND** command, for example, uses quotes to surround a character string that is being searched for. A typical use of this command in the DOS environment would be:

```
find "October" memo
```

(This command displays all lines in the file **memo** containing the word "October.") If you issue the command from the UNIX shell in this form, the UNIX shell strips away the quotes and passes the command in an illegal form to DOS. To avoid this unwanted behavior, type:[2]

```
dos find \"October\" memo
```

You can sometimes avoid unwanted interpretation of metacharacters by including DOS drive designations when you run a command. To accomplish the copying operation described above, for example, you can type:

```
dos copy c:*.com a:
```

Provided you do not have a file in your current directory that matches the character string "c:*.com" (including the "c:"), this command works as expected. When you issue this command, the UNIX shell first looks for files matching the character string "c:*com" in order to carry out the proper substitution for the metacharacter asterisk (*). Because the shell cannot find matching files, interpretation of the metacharacter does not occur. The string "c:*.com" is passed literally to DOS, which interprets the drive designation and metacharacter as expected.

Avoiding UNIX-DOS program name conflicts

There are a few UNIX programs with the same names as DOS programs. The UNIX and DOS operating systems both include **copy**, **sort**, and **find** commands, for example. The **copy** and **sort** commands operate somewhat differently in the two operating systems, and **find** has an entirely different purpose in DOS from that in UNIX. You might encounter other program names shared by UNIX and DOS. These duplicated program names create no problem when you are working inside the DOS window, where all commands are assumed to be DOS commands.

From the UNIX shell, because you can issue both DOS commands and UNIX commands, you might not be sure which version of **copy**, **sort**, or **find** is executed. By default, UNIX commands are executed instead of DOS commands because the directories containing UNIX commands are listed first in the default search path.

2. Again note that the command starts with **dos** to avoid a conflict with the UNIX **find** command.

There are three ways you can ensure that the DOS versions of these commands are executed:

1. Explicitly specify the path of the command when you run it, for example:

   ```
   /usr/dbin/copy names newnames
   /usr/dbin/sort < names
   ```

   Since */usr/dbin* contains the standard DOS commands, you know that the DOS versions of **COPY** and **SORT** are run.

2. Start your UNIX command line with **dos**, which guarantees that the command is interpreted as a DOS command:

   ```
   dos copy names newnames
   dos sort < names
   ```

3. Change your path to place the directories containing DOS commands at the beginning of the path. (This procedure has no effect on commands built into the UNIX shell, such as **type**. These built-in commands are interpreted and executed without searching the path.)

Inapplicable DOS commands

Several DOS commands have no known use from the UNIX shell. The UNIX system provides the same functions as some of these DOS commands, in many cases with commands having the same names. Other DOS commands (CD, for example) are not useful from the UNIX shell because they affect a transient DOS environment that lasts only as long as the command itself. The following DOS commands are not enabled for use from the UNIX shell:

APPEND	**CLS**	**GRAPHICS**	**NLSFUNC**	**SET**
ASSIGN	**CTTY**	**JOIN**	**PROMPT**	**SHARE**
BREAK	**FASTOPEN**	**KEYB**	**PATH**	**SUBST**
CD (CHDIR)	**FDISK**	**MKDIR**	**RECOVER**	**VERIFY**
CHCP	**GRAFTABL**	**MODE**	**RMDIR**	

All DOS batch commands except **FOR**

The following additional restrictions apply to DOS commands:

- You cannot use **CHKDSK, FORMAT, SYS, UNDELETE** or **UNFORMAT** on the shared DOS/UNIX file system. You can, however, use these commands on an actual DOS file system (such as a diskette drive or a DOS partition) as you would on any conventional DOS system.
- The DOS **TIME** and **DATE** commands can be used from the UNIX shell to display the time or date, but cannot be used to alter the DOS Services system clock. Note that the UNIX system also includes **time** and **date** commands, so if you use DOS **TIME** or **DATE**, see "Avoiding UNIX-DOS program name conflicts" (page 164).

Multitasking and background execution

Because UNIX is a multitasking system, you can run more than one program at a time. UNIX users commonly run noninteractive programs (such as compilers) in the background while simultaneously running an interactive program (such as a text editor) in the foreground. When a program is running in the background, your console or terminal is available for other tasks; you do not need to wait for the background program to complete before beginning another task.

To start a program in the background, type an ampersand (&) at the end of the command line.

You can run some DOS programs (including batch files) as well as UNIX programs in the background by invoking them from the UNIX shell prompt and including an ampersand at the end of the command line. For example, if the Microsoft C compiler is installed on SCO Open Desktop/SCO Open Server, you can use it to compile and link a file named *hist.c* in the background by typing the command:

```
cl hist.c &
```

Just as with the UNIX **cc** command, your UNIX prompt returns as soon as you issue the **cl** command, and you can continue using other UNIX or DOS programs while the DOS **cl** program compiles and links *hist.c* in the background.

NOTE You can successfully run only noninteractive programs in the background. Most DOS programs are interactive, so you cannot run most DOS programs in the background. Refer to the description of the **dos ±b** option in the *Administering DOS Services* section of the *System Administrator's Guide* or to the **dos**(ADM) manual page for further information on installing and using noninteractive DOS programs.

In DOS Services, you can use the standard DOS pipe and redirection mechanisms with DOS programs and files just as you would on a conventional DOS computer. DOS Services gives you the additional power of piping and redirecting between DOS and UNIX programs.

Using DOS drives

You can use DOS drives from the UNIX shell to access DOS files and programs stored outside the shared UNIX/DOS filesystem. Files on a diskette drive that is accessible to DOS but not mounted as part of the UNIX filesystem, for example, are not part of the shared UNIX/DOS filesystem. DOS partitions are also not part of the shared UNIX/DOS filesystem. See *Administering DOS Services* in the *System Administrator's Guide* for further information on DOS partitions.

Follow these rules to use DOS files that are not part of the shared UNIX/DOS filesystem:

1. If you want to run a command on a DOS drive, precede the command with the word "dos" and specify the drive name as you would when using standard DOS:

   ```
   dos a:cards
   ```

2. If the drive name is a parameter to a command, use the drive designation just as you would when using standard DOS:

   ```
   dir a:
   dir a:memos/july
   ```

These rules apply to any DOS filesystems, including physical DOS partitions, virtual DOS partitions, and virtual floppies. The following examples illustrate these principles:

- Run a program called **TRICKS** stored on the primary physical DOS partition:

  ```
  dos e:tricks
  ```

- Using the **dos +a** option, run a DOS program called **SHOWDATA** stored on the virtual floppy **/usr/paul/vflop** to display information about the file **messages** in your home directory (drive *D:*):

  ```
  dos +ab:=/usr/paul/vflop b:showdata d:messages
  ```

  See "Attaching devices with dos +a" (page 154), the **dos**(ADM) manual page, and *Administering DOS Services* in the *System Administrator's Guide* for more information on the **+a** option.

- Again using the **dos +a** option, display the contents of the virtual partition **/usr/sam/vpart**:

  ```
  dos +af:=/usr/sam/vpart dir f:
  ```

Stopping DOS programs

There are several ways to stop a DOS program that you started from the UNIX shell. To stop a DOS application, you should generally follow the application manufacturer's instructions. If these instructions do not stop the application, or if you are running a DOS program for which no specific procedures are provided, use one of the following procedures. You may need to try more than one of them.

1. Send the UNIX interrupt signal by pressing the ⟨Del⟩ key. An interrupt aborts only stream-oriented programs running in the foreground.

2. Use the DOS break character by pressing ⟨Ctrl⟩⟨C⟩ or ⟨Ctrl⟩⟨Break⟩.
3. Press ⟨Ctrl⟩⟨Alt⟩⟨Del⟩.
4. Use the **KILL** DOS terminal control code described in "Stopping DOS programs" (page 125).
5. If the DOS process is running in the background (and you see the UNIX prompt), use the UNIX **ps** command to determine the process ID (PID) and issue the UNIX **kill** command in the form:

   kill ***pid***

   Here, *pid* is the process ID of the DOS process you want to kill. (See "Stopping runaway processes" (page 99) for information about ps and kill.)

DOS and UNIX programs can communicate with each other in SCO Open Desktop/SCO Open Server because DOS programs (including the DOS environment) run as UNIX processes.

Printing from the UNIX shell

When you do your work at the UNIX shell, you have the option of printing either DOS or UNIX files with either DOS or UNIX print commands.

Using UNIX print commands

By default, DOS Services sends all printer output from both DOS print functions and UNIX print functions to the UNIX spooler. Using UNIX print commands is therefore usually more efficient than using DOS print commands. When you run a DOS print command at the UNIX shell, it indirectly invokes a UNIX print command, that is, a command you could run directly from the UNIX shell yourself. If you are familiar with UNIX print commands, we recommend that you use them to print both DOS and UNIX files when you use the UNIX shell.

Using DOS print commands

If you prefer to use DOS print functions, you can use most of them just as though you were running UNIX printing routines. Following are the general rules for using DOS print functions from the UNIX shell.

- The DOS **PRINT** command works in the standard way from the UNIX shell. To print, issue the print command with the name of the file you want to print:

  dos print ***filename***

  You cannot use **PRINT** options (such as /T, /C, and /P) when you use UNIX spooling.

- You can use the DOS COPY command to copy a file to the DOS print device PRN, LPT1, LPT2, or LPT3:

  dos copy *filename* lpt1

- Printing done by DOS applications works just as it does in the DOS environment. DOS Services stores the printer output in a temporary file and sends it to the UNIX spooler when either of the following occurs:
  1. You exit the application and return to your UNIX prompt.
  2. More than 15 seconds have elapsed since the application has sent a character to be printed.
- You cannot use the DOS ⟨Prt Sc⟩ and ⟨Ctrl⟩⟨Prt Sc⟩ functions from the UNIX shell.
- If you want to print a file that is not in the shared UNIX/DOS filesystem, you must use a DOS print or copy command. For example, to print files from DOS drive *A:*, you would use a command such as:

  dos copy a:status lpt1

Running UNIX programs from DOS

The DOS Services **on** utilities allow you to run UNIX programs from the DOS environment and view the output as though the programs were actually running under DOS. You can also view status information concerning UNIX programs and control their execution and output from the DOS environment.

The **on** utilities can only be used to execute *noninteractive* UNIX commands — those that do not initiate a conversation with the user. You must be at a UNIX prompt if you want to run interactive UNIX programs such as the **vi** text editor.

With the **on** utilities you can:

- Execute UNIX commands without switching to a UNIX window or screen or exiting the DOS environment.
- Extend the functionality of the DOS environment by executing UNIX commands on DOS files as if they were DOS commands.
- Run UNIX programs in the background and view their output at a later time.

The **on** utilities include three commands that run under DOS: **on, jobs,** and **kill.**

Specifying the machine and UNIX command

UNIX commands can contain all options and arguments exactly as they would be typed at a UNIX prompt. For example, from your DOS prompt:

```
on unix pr -o10 -w65 -l54 -d /tmp/longfilename1
```

Specify UNIX file names with their full UNIX names, not with their mapped DOS names.

The **on** command runs the specified UNIX process in the current directory of your current drive, provided it is a drive (such as *C:*, *D:*, or *J:*) that accesses the shared UNIX/DOS filesystem. For example, if your current working drive is drive *C:* and you type:

```
on unix rm temp
```

on removes the file *temp* in your current working directory on drive *C:*. If your current working drive is drive *A:* and you type:

```
on unix rm temp
```

on removes the *same* file as in the previous example — the file *temp* in your current working directory on drive *C:*. If your current drive is not one of these drives, **on** fails and displays an error message.

The **on** command cannot execute multiple UNIX commands separated by semicolons unless the commands are surrounded by parentheses. For example:

```
on unix (ls ; cat names)
```

The **on** command automatically converts the text output of the UNIX command from UNIX format to DOS format. That is, the DOS Services **unix2dos** utility is built in.

If **on** cannot execute the requested UNIX command, either because it cannot find a requested file or because you do not have execute permission for a requested file, then **on** returns the following error message:

```
unixcommand: access denied or file not found
```

where ***unixcommand*** is the name of the command that **on** attempted to run.

Using UNIX command names directly

The second major form of the **on** command allows you to run UNIX commands without typing **on unix** for every command. When you want to avoid typing **on unix**, copy or link the **on** program to the names of the UNIX commands you want to run directly from the DOS prompt.

The executable DOS file that contains the **on** command is *\USR\DBIN\ON.EXE*. To make a UNIX command executable directly from the DOS prompt, copy or link *ON.EXE* to a file with the name of the UNIX command you want to run. Include the file name extension *.EXE* in the renamed copy of *ON.EXE*. Assume, for example, that you have a UNIX program called **getname** that displays a user's full name when given either a first or last name. To make **getname** executable under DOS, copy *ON.EXE* with the command:

```
copy \usr\dbin\on.exe getname.exe
```

You could then type:

```
getname joe
```

from the DOS prompt, and Joe's full name is printed in your DOS window.

All options and arguments are entered following the renamed copy of **on** exactly as they would be at the UNIX prompt. This feature allows you to make copies of **on** with the names of UNIX utilities and use them as if they were DOS utilities.

Observe the following precautions and restrictions:

- The file names of your UNIX commands must have the extension *.EXE*, just like *ON.EXE* does.
- As with any DOS command in the DOS environment, the copies of *ON.EXE* must be in your DOS search path or your current directory.
- The UNIX command that *ON.EXE* is renamed to resemble must contain only lowercase letters in its name. (Other commands can still be run by preceding the command with **on unix**.)
- The name of the UNIX command cannot violate any of the DOS naming conventions. For example, you could not create an **on** version of a UNIX shell script named **mycalendar** because its name contains ten letters. (You can still run **mycalendar** by using the command **on unix mycalendar**, however.)
- Do not try to make copies of *ON.EXE* with the names of UNIX programs called **jobs** or **kill**. The **on** utilities interpret **jobs** and **kill** as built-in commands. Whenever you run these commands directly from your DOS prompt, DOS Services runs the **on** utilities versions. If UNIX versions of these commands exist and you want to execute them from the DOS prompt, use the first form of the **on** command, like this:

  on unix jobs
  on unix kill *pid*

- Do not make copies of **on.exe** with the names of DOS internal commands. The DOS shell interprets DOS internal commands, such as **TYPE**, as soon as they are entered. You therefore cannot run **on** utilities versions of these commands directly from the DOS prompt. To run a UNIX command with the same name as a DOS internal command, use the first form of the **on** command, like this:

  on unix type cat

Linking on in the shared UNIX/DOS filesystem

The **on** command is stored in *\USR\DBIN\ON.EXE* in the shared UNIX/DOS filesystem. Rather than using up disk space by making copies, you should, whenever possible, use the UNIX **ln** command to create links to the **on** program. The advantage of using the UNIX **ln** command instead of the DOS **COPY** command is that **ln** allows a single file to have more than one name without taking up extra disk space.

When DOS Services is installed, several UNIX programs are already linked to *\USR\DBIN\ON.EXE*. These programs are:

cat	df	egrep	lp	spell
chmod	diff	fgrep	ls	tail
cmp	dosopt	grep	mv	wc
cp	du	ln	pr	

Like *ON.EXE*, these links are in the directory *\USR\DBIN*. Because these links exist, you can use any of these commands in the DOS environment just as though they were DOS commands.

Search path and other environment considerations

All forms of the **on** command use both DOS and UNIX search paths. If you use the form **on unix** ***unixcommand***, DOS must be able to find the file *ON.EXE*. *ON.EXE* is in *\USR\DBIN*, which is in the default DOS search path. If you use the form ***linked_unix_command*** or ***copied_unix_command***, DOS must be able to find the file that is linked to or a copy of *ON.EXE*. With any form of the **on** command, UNIX must be able to find the UNIX command named in the **on** command line.

DOS Services executes UNIX commands that you run with **on** under the standard Bourne shell, **sh**. Any UNIX environment variables exported by the shell that started your DOS environment are available to UNIX programs executed with **on**.

Breaking out of on

Unless **on** is running as a detached task, as described in the next section, you can interrupt it by pressing ⟨Ctrl⟩⟨C⟩ or ⟨Ctrl⟩⟨Break⟩. The following prompt is displayed:

```
a - abort, c - continue, d - detach:
```

Entering **a** kills the job, clears the job from the job table, and returns you to a DOS prompt. Entering **c** allows the process to continue as before in the DOS foreground. Entering **d** detaches the UNIX process from your terminal, returns your DOS prompt, and continues to run the specified UNIX program in the background under UNIX. The next section describes the job table and detached tasks.

Running on jobs in the background

The **on** utilities allow you to:

- Start one or more UNIX commands from your DOS prompt and run them in the background.
- Detach an **on** command that is currently running in the foreground so that it continues to run in the background. Your DOS prompt returns and you can issue other commands while your **on** program continues.
- View a table showing the status of all currently executing and completed **on** commands.
- View the output of detached commands, including commands that have finished executing, at any time.

The following sections describe the features for controlling jobs executed with **on**.

Using the job table

DOS Services maintains a job table that keeps track of detached UNIX processes initiated by **on** — that is, processes that are placed in the background of the UNIX environment. The job table shows the status of up to ten detached **on** commands. Each DOS environment has its own job table.

The job table holds one entry per **on** command. Each UNIX command, however, may create several subordinate processes. For example, a single **spell** command takes only one space in the DOS job table, but appears in a UNIX **ps** (process status) list as four or five processes.

Two ways to detach tasks

You can detach UNIX tasks initiated with **on** so they run in the UNIX background in either of two ways:

1. Add an ampersand (&) to the end of the **on** command. For example:[3]

   spell memo &

   Note that the ampersand must be preceded by a space. The following command is incorrect:

   spell memo& *(incorrect)*

2. Initiate an **on** command and later interrupt it with ⟨Ctrl⟩⟨C⟩ or ⟨Ctrl⟩⟨Break⟩. For example:

   spell memo
   ⟨Ctrl⟩⟨C⟩

   The following prompt is displayed:

   ```
   a - abort, c - continue, d - detach
   ```

   When you enter **d**, the task is placed in the background under UNIX.

In either case, **on** responds with a message in the form:

[*jobnumber*] *pid*

where *jobnumber* is the job number of the task in the DOS job table, and *pid* is the UNIX process ID number returned by UNIX. Your DOS prompt then returns and you can issue additional commands (including **on** commands) while your detached process continues executing in the background. Any output produced by a detached **on** task (for example, the spelling errors found by the **spell** command shown previously) is by default stored in a temporary file. You can see the contents of this temporary file by reattaching the task as described in "Reattaching to detached tasks" (page 176). You can also use pipes and redirection to send the output of detached tasks to any file or program you choose. See "Using pipes and redirection" (page 177).

3. This example and all subsequent examples of standard UNIX commands in this chapter use the second **on** command form. That is, the words **on unix** are not explicitly included in the command since the names of these UNIX commands are linked to *\USR\DBIN\ON.EXE*. All of these examples work equally well if you include **on unix** at the beginning of each command.

Keeping track of detached tasks

You can use the **jobs** command to perform any of the following tasks:

- Display job table information.
- Clear the job table of entries for completed jobs.
- Reattach to detached jobs.

Invoking **jobs** with no arguments displays the current job table in the following format:

```
JOB     STATE          EXIT STATUS    COMMAND
[1]     Running                       unixcommand1
[2]     Done           exit(0)        unixcommand2
```

If the job table is currently clear, invoking **jobs** returns you to the DOS prompt. The job table is clear when no detached jobs have yet been run or when you have cleared the job table of all entries.

The four columns in the job table report the following information:

1. The "JOB" column shows the job ID number that **on** assigned to each process when the process was detached.
2. The "STATE" column indicates whether the process is running or done.
3. The "EXIT STATUS" column shows one of the following values:

`exit(`***nn***`)`	The job terminated with an exit status of ***nn***. An exit status of 0 usually means the process terminated normally. Any other value may indicate an abnormal termination of the process.
`unknown`	The job terminated, but **on** was unable to determine its exit status. This condition does not normally occur.
`signal(`***nn***`)`	A signal was received that killed the process. In this case, ***nn*** indicates the signal received.
`coredump(`***nn***`)`	The signal received caused a core dump to occur. This is a special case of **signal**.
`err3(`***nn***`)`	An error in the functioning of **on** or DOS Services has occurred.

4. The "COMMAND" column shows the UNIX command that was requested.

Completed processes remain in the job table until you clear them by entering **jobs** with a single hyphen as an argument. When you enter:

jobs -

these events occur:

- The current status of the job table is displayed, including all currently "Done" entries.

- The "Done" entries are cleared from the job table.
- Any temporary files in the *\TMP* directory that were associated with the "Done" jobs are removed.

Once jobs are cleared from the job table, they can no longer be reattached. Output from any task being saved in temporary files for reattachment and review is discarded.

Reattaching to detached tasks

Reattaching to a detached task allows you to:

- View all previous output of a currently running program and continue to view new output as the program sends it to your screen.
- View the output of a program that has finished running.

To reattach either to a currently running job or to a completed job, use the **jobs** command in the form:

jobs %*jobnumber*

where ***jobnumber*** is the job number of the detached task.

For example, suppose you start the UNIX **spell** program as a detached task by typing:

spell memo &

When you type this command, **on** displays a job number and process ID similar to:

[1] 4376

To reattach to your **spell** job, type the command:

jobs %1

If you decide to reattach to the detached job and do not know the job number, type the **jobs** command to find out. A display similar to this is printed:

```
JOB     STATE            EXIT STATUS     COMMAND
[1]     Running                          spell memo
[2]     Done             exit(0)         who
```

Using the percent sign without a number reattaches you to the lowest numbered task in the job table:

jobs %

When you reattach to a currently running job (that is, one that appears as "Running" in the job table), you see all the output produced by the job up to the time you reattach to it. The job then continues to run, and you see any additional output as it is printed. If you wish, you can detach the task again at any time while it is running. Each time you reattach it, you see *all* output printed up to the time you reattach.

When you reattach to a completed job (one that appears as "Done" in the job table), you see all output printed by the job. The temporary file storing the output is removed and the job table entry for that job is cleared. The temporary file is also removed and the job table entry cleared if the job completes while it is attached.

Stopping detached jobs

You can halt detached jobs and clear them from the job table by reattaching, pressing ⟨Ctrl⟩⟨C⟩ or ⟨Ctrl⟩⟨Break⟩, then responding a (for "abort") to the resulting prompt.

You can also stop a UNIX process by using the **on** utilities **kill** command, which runs under DOS, in this form:

kill [-*signal*] %*jobnumber* [. . .]

where ***jobnumber*** is the task number of the job in the job table and ***signal*** is the UNIX signal to be sent to the process.

The **on** utilities **kill** command works exactly like the UNIX **kill** command except that it accepts ***%jobnumber*** instead of the process ID number argument. See the descriptions of **kill** and **signal** in your UNIX documentation for more information. The default signal sent is 15. A signal of 9 may be used for a sure kill. Note, however, that entering **kill -9** is bad practice unless other **kill** commands have failed, because the UNIX program has no chance to perform cleanup operations before exiting.

Using pipes and redirection

With the **on** command, you can use either DOS or UNIX pipes or redirection, or even combine the DOS and UNIX versions of these mechanisms in a single command. You can accomplish nearly all useful operations using the DOS mechanisms. You can use DOS pipes (|) with UNIX programs that you invoke using **on** just like you use them with DOS programs and you can redirect the output of **on**-initiated commands to specified files. or directories where you have write permission. You can also use the DOS input redirection mechanism (<) to redirect standard input from a file or from the DOS keyboard to an **on** command.

Input redirection from a file can be useful when you want to run a UNIX command that operates on a file contained on a DOS device, such as drive *A:*. For more information on using DOS pipes and redirection utilities, see you DOS documentation. For more information on UNIX pipes and redirection utilities, see "Entering UNIX commands" (page 67).

Using pipes and redirection in detached tasks

A detached **on** command that includes DOS pipes or redirection may not produce the results you want. This can happen because the DOS command processor, *COMMAND.COM*, interprets the pipe and redirection symbols (<, >, and |), while **on** interprets other parts of the command line, including the ampersand (&).

For example, if you want to redirect the output of the UNIX **spell** command into a file called *TYPOS*, you might incorrectly issue the following command:

```
spell memo > typos &          (incorrect)
```

This command, however, does not put the list of spelling errors in the file *TYPOS*. Instead, it puts the job number and process ID of the **spell** command (which **on** returns to the DOS environment) in *TYPOS*.

To save output produced by **spell** you can do either of the following:

- Issue the command in the form:

  ```
  spell memo &
  ```

  and then redirect the output when you reattach, as described previously under "Saving output from completed jobs."
- Use UNIX output redirection rather than DOS output redirection when you issue the **spell** command. You can also use quotes to prevent the pipe and redirection symbols from being interpreted by DOS. For example:

  ```
  spell "memo > typos" &
  ```

  This command, however, leaves the contents of the file *typos* in UNIX text format.

The **on** utilities use the special characters " { " to mean UNIX input redirection, " } " to mean UNIX output redirection, and " ! " to mean a UNIX pipe. When you use these characters in an **on** command, the UNIX system does the redirection or piping operation, and the results are often easier to understand. To use UNIX output redirection in the **spell** example illustrated above, type:

```
spell memo } typos &
```

This example displays a job number and process ID when you issue the command. Your DOS prompt returns, as expected, and the task runs in the background. When the **spell** program completes, the results are in the file *TYPOS*.

When you use UNIX redirection as illustrated in the previous example, the output of the **on** command is not returned to the DOS environment and is not converted into DOS text format. You can use the DOS Services **unix2dos** command to convert the file to DOS text format.

The following example shows how UNIX pipes and redirection can be useful. It shows both incorrect and correct ways of creating a file called *NAMES* containing a sorted list of users currently logged into the DOS Services system.

```
on unix who | on unix sort > names &            (incorrect)
on unix who ! sort ! unix2dos } names &(correct)
```

If you issue the first command, it is not detached until the **who** program finishes, and the file *NAMES* contains a job number and process ID rather than a sorted **who** list. This is probably not the result you want. The second command uses a UNIX pipe to send the list output by the **who** program directly to the UNIX **sort** program, without returning to DOS or using a second **on** command. Because the output is not returned to the DOS environment, it remains in UNIX text format rather than being converted to DOS text format. The **unix2dos** program is therefore used to convert the list to DOS format, following which UNIX output redirection (}) puts the list into the file *NAMES*.

To be properly interpreted by **on**, the special redirection and pipe characters ({, }, and !) require spaces surrounding them. The following examples, which omit the required spaces, are incorrect:

```
spell memos}typos &                             (incorrect)
on unix who!sort!unix2dos}names &               (incorrect)
```

Chapter 19 Using Windows with DOS Services

DOS Services supports two versions of Microsoft Windows, Windows 3.0 and Windows 3.1. The differences between these two programs are significant and affect the kind of support DOS Services offers.

Windows 3.0 support

Windows 3.0 is supported in the 8086-compatible "Real Mode." It offers access to no more than 1 Mbyte of RAM. Windows 3.0 and Windows applications that run under it will operate only in full screen VGA mode. If you run Windows 3.0 in an X Window, that window must be zoomed to full screen size.

If you want to provide Windows 3.0 with memory beyond the standard 1 Mbyte limit, you must attach Expanded Memory (as opposed to Extended Memory). This is done with the DOS Options Dialog Box (see "Using the DOS options dialog box" (page 151)) or the **dos +a** command. For more information on the **dos +a** command see "Attaching devices with dos +a" (page 154) or the **dos**(ADM) manual page.

Windows 3.1 support

Windows 3.1 is supported in a very different manner. Special modifications have been made to DOS Services to support Protected Mode operation, access to Extended Memory, and Desktop (X Client) operation.

Version 3.1 of Windows runs under DOS Services on an emulated 80286 processor; it operates in the Standard or 286 Protected Mode. It multitasks Windows applications and can give them access to up to 15 Mbytes of RAM. It operates as a true X Client, running properly within a zoomed or an unzoomed X Window thus making it possible to function on your personal Desktop.

Installing Windows 3.1

Installing Windows 3.1 under DOS Services is very similar to installing Windows 3.1 on native DOS. This section describes the steps you need to perform. If you have never installed Windows before, you should review the installation description in your Windows documentation.

Windows is installed by a DOS program called **SETUP**.

SETUP copies two versions of itself to your Windows directory along with the rest of the Windows files. The first time you run **SETUP** you invoke it from a diskette, usually in drive *A:*. After that it is available from your Windows directory.

SETUP is used not only to install, but also to reconfigure Windows. One version runs from the DOS command line. The other operates from inside the Windows environment, accessed through the Windows Setup icon in the Main Group.

SETUP is intelligent enough to sense whether it is being run from a diskette (to install) or from the Windows directory (to reconfigure) and to do the appropriate thing.

Installing a personal copy of Windows 3.1

1. To start the appropriate DOS environment, double click on the WinSetup icon, which is in the Accessories folder of your Desktop.
2. When you get your DOS prompt:
   - Insert the first disk of the Windows 3.1 package in drive *A:* (You can also install Windows 3.1 from drive *B:*).
   - Enter the command:

     ```
     A:SETUP
     ```

   If the text is not readable, press ⟨Alt⟩⟨D⟩ to bring up the DOS menu. Select the DOS **Colors** option.
3. The Windows 3.1 **SETUP** program checks your environment and prints a warning about the use of the **SUBST** in your DOS session. Press **C** to continue.
4. **SETUP** checks for previous versions of Windows on your hard disk. This may take a few minutes. If it takes more than three minutes, see
5. Later displaying a welcoming screen, Windows prompts you to choose between the **Express Setup** and the **Custom Setup**. For proper operation in an DOS Services environment, we recommend the **Custom Setup**.

6. **SETUP** then displays a suggested disk and directory name to be used in installation. Generally, this is *C:\Windows.* If Windows has been installed previously, the suggested directory will be the same as the installed copy. For correct operation, DOS Services requires that Windows be installed on your *D:* drive. Change the displayed disk/directory to the *D:* drive, and a directory name of your choice (for example, *WIN31* or *WINDOWS*).

7. **SETUP** displays the current hardware configuration (type of video, mouse, etc.). Verify that the displayed hardware is correct. DOS Services always sees the mouse as a two-button, Microsoft-compatible mouse. Do not change this option even if your actual hardware is different. If any other part of the displayed configuration is incorrect, use the arrow keys to move the cursor to that field and make any correction needed. On some systems, for instance, **SETUP** may mistake your 101-key keyboard for another keyboard. When the display is correct, move the cursor to the option:

   No changes: The above list matches my computer

   and press enter.

8. **SETUP** copies some files and invokes Windows. At this point you must Zoom to full screen before you can continue with your installation. To Zoom, invoke the DOS Menu by pressing ⟨Alt⟩⟨D⟩ and click on the "Zoom" button.

9. **SETUP** prompts you to enter your name and the name of your company.

10. **SETUP** displays a screen with three installation options with an X in each options check box. The options are

    Setup only the Windows Components you select
    Setup Printers
    Setup applications installed on your hard disk

    Delete the check marks next to any of these options you wish to bypass. Although all of these options will work under DOS Services, you can install faster by deleting the check marks for the first and third option. If you don't want to install a printer, you should also delete the check mark next to the second option. (In the case of limited disk space, you might want to use the first option.)

11. You may get an error message about problems updating your system files. If so, select the **Cancel** option to continue the installation.

12. **SETUP** may display a request to modify your *AUTOEXEC.BAT* and *CONFIG.SYS* files. Do not let these files be modified - pick the option that says you will make modifications later.

13. **SETUP** now displays a list of printers (assuming you selected that option). Install the appropriate printers, making sure you pick *LPT1.DOS* for the port.

14. **SETUP** creates your Windows desktop and offers to run a tutorial. Click on the Skip Tutorial button so that you can finish installation. (You can run the tutorial at a later time.)
15. **SETUP** prompts to stay in Windows or return to DOS. Click on the DOS button.
16. You must now add the Windows path to your DOS **PATH** environment variable. You should do this in your local startup file, *D:\AUTOEXEC.BAT* If you have a local *AUTOEXEC.BAT* file in the root of your *D:* drive, update the **PATH** entry to include the Windows path. If no *AUTOEXEC.BAT* file exists on your *D:* drive, create one with a DOS editor and add the Windows path. (Under DOS Services, your DOS **PATH** combines the **PATH** entry in the system startup file (*C:\AUTOEXEC.BAT*) plus the **PATH** entry in your local startup file (*D:\AUTOEXEC.BAT*).
17. After installation is complete, you can exit the DOS session by typing **QUIT** at the DOS prompt.

Installing the network version of Windows 3.1

With a network installation, a single binary copy of Windows can be executed by multiple users, saving disk space. Of course, you need the appropriate license agreement.

The network installation of Windows is a two step process. First, the administrator installs the Windows files into a shared directory by running **SETUP/A**. Then each user accesses the files in this shared directory to create a personal copy of Windows in a personal directory using **SETUP /N**. Only a few configuration files are actually copied to each user's personal Windows directory. The majority of the Windows files remain in the shared directory. Therefore each user must have **PATH** access to two Windows directories, the user's own personal Windows directory and the shared Windows directory.

1. To install the network version, log into your system as root and start the appropriate DOS environment by double clicking on the WinSetup icon, which is in the Accessories folder of your Desktop.
2. When you get your DOS prompt:
   - Insert the first disk of the Windows 3.1 package in drive *A:*
   - Enter:

     ```
     A:SETUP /A
     ```

   If the text is not readable, press ⟨Alt⟩⟨D⟩ to bring up the DOS menu. Select the DOS Colors option.

3. **SETUP** prompts you to enter a group and company name.
4. **SETUP** prompts for the desired Windows directory. You should install on drive *J:* (for example,*J:\WINDOWS*). This will place the Windows files in their own directory within the public-access area of the shared DOS/UNIX file system. Remember that under DOS Services, *J:\WINDOWS* and *C:\USR\LDBIN\WINDOWS* are the same directory.
5. **SETUP** prompts for the Windows diskettes and copies the contents into the shared directory.
6. When installation is complete, **SETUP** will exit.
7. Next, you need to copy some of the Windows/X driver files supplied by DOS Services into this shared directory. These are the files that allow DOS Services to run as an X Client on your Desktop. DOS Services includes a command called **WINXCOPY** to do this copying. It is automatically installed in *C:\USR\LDBIN*, which should be in your DOS search path. Unless you have changed your **PATH** statement, you can run the following command from any drive or directory. Just be sure you specify as an argument the proper drive and directory where the public-access copy of Windows is installed.

   WINXCOPY *J:\WINDOWS*

8. The network installation is now complete, and you can exit your DOS session by typing **QUIT** at the DOS prompt.

Users who want to run this shared copy of Windows must copy a few files to their own personal Windows directory. Each such user must perform the following steps:

1. Log into your system as yourself and start the appropriate DOS environment by double clicking on the WinSetup icon, which is in the Accessories folder of your Desktop.
2. When you get your DOS prompt:
   - Move to the drive and directory where the public-access copy of Windows is installed. If it installed on *J:\WINDOWS*, type the following commands:

     J:
     CD *\WINDOWS*

   - Run the **SETUP** program in that directory by typing:

     SETUP /N

3. If the text is not readable, press ⟨Alt⟩⟨D⟩ to bring up the DOS menu. Select the DOS Colors option.

4. The Windows 3.1 **SETUP** program will check your environment and print a warning about the use of **SUBST** in your DOS session. Press **C** to continue.
5. **SETUP** will check for previous versions of Windows on your hard disk. This may take a few minutes. If it takes more than 3 minutes, see "Slow installations " (page 192)
6. After displaying a welcoming screen, Windows prompts you to choose between the **Express Setup** and the **Custom Setup**. For proper operation in an DOS Services environment, we recommend the **Custom Setup**.
7. **SETUP** then displays a suggested disk and directory name to be used in installation. Generally, this is *C:\WINDOWS*. If Windows has been installed previously, the suggested directory will be the same as the installed copy. For correct operation, DOS Services requires that Windows be installed on your *D:* drive. Change the displayed disk/directory to the *D:* drive, and a directory name of your choice (for example, *WIN31* or *WINDOWS*).
8. **SETUP** displays the current hardware configuration (type of video, mouse, etc.). Verify that the displayed hardware is correct. DOS Services always sees the mouse as a two-button, Microsoft-compatible mouse. Do not change this option even if your actual hardware is different. If any other part of the displayed configuration is incorrect, use the arrow keys to move the cursor to that field and make any correction needed. On some systems, for instance, **SETUP** may mistake your 101-key keyboard for another keyboard. When the display is correct, move the cursor to the option:

   No changes: The above list matches my computer

   and press enter.
9. **SETUP** copies some files and invokes Windows. At this point you must Zoom to full screen before you can continue with your installation. To Zoom, invoke the DOS Menu by pressing ⟨Alt⟩⟨D⟩ and click on the Zoom button.
10. **SETUP** prompts you to enter your name and the name of your company.
11. **SETUP** displays a screen with three installation options, with an X in each options check box. The options are

    Setup only the Windows Components you select
    Setup Printers
    Setup applications installed on your hard disk

    Delete the check marks next to any of these options you wish to bypass. Although all of these options will work under DOS Services, you can install faster by deleting the check marks for the first and third option. If you don't want to install a printer, you should also delete the check mark next to the second option. (In the case of limited disk space, you might want to use the first option.)

12. You may get an error message about problems updating your system files. If so, select the **Cancel** option to continue the installation.
13. **SETUP** displays a request to modify your *AUTOEXEC.BAT* and *CONFIG.SYS* files. Do not let these files be modified - pick the option that says you will make modifications later.
14. **SETUP** now displays a list of printers (assuming you selected that option). Install the appropriate printers, making sure you pick *LPT1.DOS* for the port.
15. **SETUP** creates your Windows desktop and offers to run a tutorial. Click on the Skip Tutorial button so that you can finish installation. (You can run the tutorial at a later time.)
16. **SETUP** prompts to stay in Windows or return to DOS. Click on the DOS button.
17. You must now add the Windows path to your DOS **PATH** environment variable. Remember that you have two Windows paths, the shared Windows directory on the *J:* drive and your personal Windows directory on the *D:* drive. Your DOS **PATH** must include both, and your personal Windows directory must be specified first. Add these paths to your local startup file, *D:\AUTOEXEC.BAT*

    If you have a local *AUTOEXEC.BAT* file in the root of your *D:* drive, update the **PATH** entry to include the Windows path. If no *AUTOEXEC.BAT* file exists on your *D:* drive, create one with a DOS editor and add the Windows path. (Under DOS Services, your DOS **PATH** combines the **PATH** entry in the system startup file (*C:\AUTOEXEC.BAT*) plus the **PATH** entry in your local startup file (*D:\AUTOEXEC.BAT*).
18. After installation is complete, you can exit the DOS session by typing **QUIT** at the DOS prompt.

Setting Permissions in \USR\LDBIN

It is standard UNIX practice to set the permissions on directories where important system files are stored to read-only. This prevents users from damaging valuable resources, unintentionally or otherwise. */usr/ldbin* and its subdirectories, where DOS program files are stored, are usually maintained in this state.

The shared network copy of Windows will usually be installed in */usr/ldbin/windows*. Valid DOS Services names for this directory are *C:\USR\LDBIN\WINDOWS* and *J:\WINDOWS*. This directory would ordinarily have its permissions set to read-only, but this may cause problems with the installation of some Windows applications.

When users try to install applications to run under Windows, they need write permission to this directory or their installation fails. Even though each user runs a personal copy of Windows, stored in that user's own directory, the DOS or Windows application files must be written to the shared Windows directory. The read-only status prevents files from being written and the user's installation attempt aborts with a message like this:

```
Cannot write file filename to
C:\USR\LDBIN\WINDOWS\FILENAME
```

To correct the problem, the System Administrator has three choices:

- Change the directory permissions permanently, using the following procedure:
  1. Change the permissions on the directory */usr/ldbin/windows* by issuing the following command

     # chmod 777 /usr/ldbin/windows

  2. Change directories into */usr/ldbin/windows*
  3. Change the permissions on all the files in this directory to "644".

     NOTE You cannot change them by issuing the command:

     chmod 644 *

     This is because there are to many files for the command to change at once. Use the **chmod** command to change the file permissions in chunks.

  This allows users access for installation.
- Change the directory permissions temporarily. Notify all users that part of their installation routine is to advise the System Administrator. The System Administrator sets directory permissions to read-write temporarily and sets them back to read-only when the installation is done.
- Take over the installation of shared Windows applications as a System Administration chore.

Each System Administrator must decide which policy works best for his or her installation.

Installing Windows 3.1 on a DOS partition

If you have a primary physical DOS partition on your fixed disk, it has a dual personality that can hinder Windows operation. For the purpose of the following discussion, let us assume that there are two basic ways of starting DOS. You can start DOS under UNIX, or you can boot DOS directly from a primary DOS partition.

When you start DOS Services from UNIX, either from your Desktop or from the UNIX command line, the primary physical DOS partition is automatically made available to your DOS session as drive *E:*. When you boot DOS directly from the DOS partition, that same section of disk is recognized as drive *C:*. This can cause problems for any DOS program which stores drive and directory information needed for operation. The basis of the problem is a drive letter mismatch. Windows installs the item under one drive name and tries to access it when the drive is called something else.

When you install Windows or configure applications to run under Windows, Windows stores drive and directory information in its initialization files, *FILENAME.INI*. For the purpose of this discussion, assume that you install Windows and its applications in a single directory called *\WINDOWS*, which is located immediately off the root of the primary DOS partition.

If you install Windows under DOS Services, the drive is labeled *E:*. The location of items like Windows' program groups is recorded in *PROGRAM.INI*. The lines look like this:

```
Group1=E:\WINDOWS\GAMES0.GRP
Group2=E:\WINDOWS\GAMES1.GRP
```

Similarly, if you configure a DOS application program to run under Windows, and you perform that configuration process under DOS Services, the program's startup information is recorded in a line that looks something like this:

```
E:\WINDOWS\DOSAPP.PIF=1 8 12 648 328 22 0 1 .......
```

In all the cases above, drive letter *E:* is part of the string which represents the item's location.

If you now boot DOS from the DOS partition, that same disk is labeled *C:*. The items above must be accessed through strings that contain the drive letter *C:* and look like this:

```
C:\WINDOWS\GAMES0.GRP
C:\WINDOWS\GAMES1.GRP
C:\WINDOWS\DOSAPP.PIF
```

The items themselves have not moved, but they must now be accessed through a different drive letter. Windows examines its initialization files for the things it needs to load properly and searches for them on the specified drive. Since the drive is now labeled *C:*, the path does not properly reflect the location of the file. The result is a failure to load.

CAUTION No matter which way you install Windows, under DOS Services or under "raw" DOS booted directly from the DOS partition, if you then boot DOS in the other manner and invoke Windows, Windows will fail.

The solution is simple. Install the items under a single drive letter and always access them under that same letter. This can be done by "renaming" the drive through a DOS **SUBST** command. This is simple to do under "raw" DOS and much more difficult under DOS Services, so the **SUBST** should always be applied when booting directly from the DOS partition.

There are two ways to do it. You can install the programs under DOS Services and then rename the drive when you run them from "raw" DOS. Or, you can boot "raw" DOS, rename the drive, perform the installation, and then run programs under DOS Services.

Installing under DOS Services

Install Windows under DOS Services as described above. Have **SETUP** place the files in *E:\WINDOWS*. Windows will then run properly under DOS Services.

When you boot DOS directly from the DOS partition, use a **SUBST** command to rename the drive. Place a line like the following in the *AUTOEXEC.BAT* on the DOS partition so that it is invoked every time you boot from the partition:

```
subst e: c:\
```

Make sure that *HIMEM.SYS* and *SMARTDRV.SYS* are in the root directory. If they are not, find them and copy them there.

Open the *CONFIG.SYS* file with an editor and look for lines like this:

```
device=e:\himem.sys
device=e:\smartdrv.sys
```

Remove the drive specification from these lines so that they look like this:

```
device=himem.sys
device=smartdrv.sys
```

The *CONFIG.SYS* file is read before the **SUBST** command in *AUTOEXEC.BAT* can be carried out. Moving *HIMEM.SYS* to the root and removing the drive specification from its *CONFIG.SYS* entry assures that it will be found at bootup time, no matter what drive letter is in force. Do the same with any other drivers that *CONFIG.SYS* must load.

Reboot and test your installation.

Installing under raw DOS

Some Windows applications install more easily under "raw" DOS than they do under DOS Services. If you run into such an application, you may wish to switch the process around.

1. Boot "raw" DOS from the DOS partition.

2. Be sure you have the DOS command **SUBST.EXE** available on that drive and in the path that will actually be searched when *AUTOEXEC.BAT* is run. One simple way to assure this is to copy **SUBST.EXE** to the root directory of the DOS partition.
3. Enter the following DOS **SUBST** command from your DOS prompt:

   SUBST *E: C:\*

4. Move to *E:* by typing:

   E:

   Notice that your prompt now reads `E:`.
5. Place the first Windows **SETUP** disk in drive *A:* and enter the command:

   A:SETUP

6. Continue with the steps as outlined above under "Installing a Personal Copy of Windows 3.1." **SETUP** complains about the use of a **SUBST** command, but the installation works correctly.
7. When **SETUP** prompts for the drive and directory in which to install Windows, be sure it is *E:\WINDOWS*.
8. Conclude the Windows setup procedure as described in the preceding sections.
9. Place the following line in the *AUTOEXEC.BAT* on the DOS partition. Be sure it is at the top of the file, before your **PATH** statement or any other commands. If you place it anywhere else, the commands that follow will fail with an "Invalid path" message.

   ```
   SUBST E: C:\
   ```

10. Make sure that *HIMEM.SYS* and *SMARTDRV.SYS* are in the root directory. If they are not, find them and copy them there. If they are available, be sure you use the updated Windows versions (usually found in your Windows directory) rather than the DOS versions (usually found in your DOS directory).
11. Most people who run Windows use both *HIMEM.SYS* and *SMARTDRV.SYS*. The following steps assume you want both. Open the *CONFIG.SYS* file with an editor and look for lines like these:

    ```
    device=e:\himem.sys
    device=e:\smartdrv.sys
    ```

    If you find such lines, remove the drive specification so that they look like the example below. If there are no such lines, add lines that look like these:

    ```
    device=himem.sys
    device=smartdrv.sys
    ```

The *CONFIG.SYS* file is read before the **SUBST** command in *AUTOEXEC.BAT* can be carried out. Moving *HIMEM.SYS* and *SMARTDRV.SYS* to the root and removing the drive specification from their *CONFIG.SYS* entries assures that they will be found at bootup time, no matter what drive letter is in force. Do the same with any other drivers that *CONFIG.SYS* must load.

12. Reboot and test your installation. If your mouse does not function properly, see "Switching mouse drivers" (page 197).

Your Windows is now installed to run properly, either under DOS Services or under DOS booted directly from the partition.

Follow the same general procedure to install other DOS or Windows applications on the partition. Boot "raw" DOS, change the drive identifier with a **SUBST** command and proceed with the installation. Always run under the same **SUBST** command when you boot from the partition.

This technique may be useful for any DOS program you find difficult to install under DOS Services.

Updating an existing installation of Windows 3.1

There is one situation to which the above procedures cannot be applied. The previous methods work by convincing Windows that it has been installed on drive *E:*. This allows it to work under DOS Services or under DOS booted from the partition. However, if Windows has already recorded its installation location as drive *C:*, that notation cannot be changed.

If you installed Windows 3.1 on your DOS disk before you installed DOS Services, you must reinstall it. Windows was probably installed on a drive other than *E:*. That drive information is embedded not only in ASCII files, which you can change with a text editor, but in Windows' Group Identification files, which are semi-binary in format. The only tool you have to change those files is **SETUP**.

Slow installations

Sometimes installing Windows under DOS Services seems to take an inordinate amount of time. Occasionally **SETUP** may report that it is searching for other installed copies of Windows and then hang. If you encounter this phenomenon, there is a quick solution.

As part of its setup routine, Windows checks carefully for previously installed versions. **SETUP** checks every directory on every drive, looking for another copy of *WIN.COM*. If your available DOS drives include large UNIX file systems, especially those with NFS mounted subtrees, this checking can go on for a very long time.

The quick fix is to abort the setup, create an imitation *WIN.COM* that *SETUP* can find quickly, and then start the installation over.

Abort the bogged installation and start DOS. Create the directory where you want Windows to be installed. We recommend *D:\WINDOWS*. From *D:\*, create that directory by typing:

MKDIR *D:\WINDOWS*

Copy any simple text file to the name *WIN.COM* within that directory. This command is illustrated below, using the file *CONFIG.SYS*.

COPY *C:\CONFIG.SYS D:\WINDOWS\WIN.COM*

Now you can run **SETUP** again. Start the appropriate DOS environment by double clicking on the WinSetup icon in your Accessories folder. Follow the procedures described in the previous sections.

When **SETUP** prompts for a drive and directory in which to install, give it the drive and directory where the counterfeit WIN.COM resides. **SETUP** looks there first, finds what it wants, and discontinues the search.

SETUP then overwrites the imitation with the real *WIN.COM* and the rest of the installation proceeds normally.

Configuring your Windows session

Assuming that Windows 3.1 has been installed as described above and that your **PATH** has been properly updated, run Windows by double clicking on the Win icon, which is in your Accessories folder.

If the Windows directory is in your path, You can also run Windows from your UNIX prompt with

```
$ win
```

By default, this starts a 3 Mbyte DOS session and then runs Windows. If the default configuration is not adequate, you can configure your Windows session just as you can configure your DOS session. For example, you can change the default amount of memory allocated, to a Windows session, you can set Windows to run under X on your Desktop, or you can change the number of attached DOS drives.

These configurations can be accomplished in a number of ways:

1. From the Desktop, use the DOS Options Dialog Box to reconfigure your Windows environment. To access this menu, position your mouse pointer over the Win icon and double click with your middle mouse button. (If you have a two button mouse, double click with both buttons simultaneously.)

When the DOS Options Dialog Box appears, make the appropriate changes. Click on "Start." Your Windows session will come up with the items you have selected available.

If you want to save a configuration for future Windows sessions just click on "Apply" Then double-click the Win icon to start your session.

NOTE You must first create a *.winstart* and *.dosstart* file in your *$HOME* directory before the following examples will work. One way to create those files is to click on the apply button in the DOS Option Dialog Box. (You will have to do it twice. Once in the DOS Options Dialog Box for Windows and once for DOS) For more information on the DOS options dialog box, see "Using the DOS options dialog box" (page 151).

The DOS Options Dialog Box is explained further in "Using the DOS options dialog box" (page 151).

2. You can also override the default settings from the UNIX command line. For example, to start a Windows session with 5 Mbytes of memory, enter:

   ```
   win +m5
   ```

   For more information on the **win** command, see the **win**(ADM) manual page

3. You can also use the **dosopt** command to configure your Windows environment. To display your current Windows environment, execute the following in a ScoTerm window:

   ```
   dosopt +v win
   ```

   For example, to change the default memory assignment to to 4 Mbytes, enter:

   ```
   dosopt +m4 win
   ```

Running Windows 3.1 on the Desktop (under X)

DOS Services provides a special Windows display driver that allows users to run Windows 3.1 under X. With this driver installed, users can run Windows on their Desktop like any other X application. You can iconify your Windows session, raise it, lower it, or move it around your Desktop.

The Windows/X driver is provided as a standard Windows display driver. It is installed using the standard Windows procedure, as if you were attaching a new display adapter. This involves running **SETUP** to tell Windows where to find the driver for the new "display card" that has been "installed."

There are two versions of **SETUP** One has a Windows graphics base and operates from within the Windows environment, accessed through an icon in the Main group. The other has a DOS text base and operates from the DOS command line. You can use either of the two.

Installing the Windows/X driver from the Windows environment

While in Windows 3.1, go to the "Main" group window and double click on the "Windows Setup" icon.

1. Pull down "Options" and select "Change System Settings."
2. Click on "display" and then select "Other display (Requires disk from OEM)"
3. **SETUP** prompts for the OEM disk to be inserted into drive *A:*. Do not insert a diskette. The files **SETUP** requires have already been installed. All you need to do is tell **SETUP** where to find them. Change the directory listed from *A:\* to *C:\USR\LIB\MERGE\WINDOWS* and press enter.
4. **SETUP** displays the following options

   DOS Merge Windows/X
   DOS Merge Windows/X (Large Fonts)

   If you are running X on a high resolution display (1024x768), choose the Large Fonts options. This option does not affect the size of your windows. It merely makes text easier to read. On any other display, choose the first option.
5. When the driver has been loaded you can restart Windows. Your screen will automatically unzoom and Windows will start under X.

Installing the Windows/X driver from the DOS environment

1. From the Desktop, start the appropriate DOS session by double clicking on the WinSetup icon.
2. Change your current directory to the directory where Windows is installed and enter:

   A:SETUP

   If the text is not readable, press ⟨Alt⟩⟨D⟩ to bring up the DOS menu. Select the DOS Colors option.
3. **SETUP** displays your current hardware configuration. Use the arrow keys to go to the "Display" option and press enter to see the list.

4. Select the option "Other display (Requires disk from OEM)" at the bottom of the list.
5. **SETUP** prompts for the OEM disk to be inserted into drive *A:*. Do not insert a diskette. The files **SETUP** requires have already been installed. All you need to do is tell **SETUP** where to find them. Change the directory listed from *A:\* to *C:\USR\LIB\MERGE\WINDOWS* and press enter.
6. **SETUP** displays the following options:

   DOS Merge Windows/X
   DOS Merge Windows/X (Large Fonts)

   If you are running X on a high resolution display (1024x768), choose the Large Fonts options. This option does not affect the size of your windows. It merely makes text easier to read. On any other display, choose the first option.
7. **SETUP** loads the driver and exits. You can now exit your DOS session by entering **quit**. You can now run Windows under X by double clicking on the Win icon.

Switching between VGA and X

The standard VGA driver provided with DOS Services has two advantages. First, it looks like "raw" DOS and some VGA graphics programs look best that way. Second, it allows some display-intensive programs to run faster. It also has one major disadvantage; it runs only in full screen mode (640x480).

The Windows/X driver runs at a much higher resolution. It is also much more flexible. It allows you to iconify your Windows session and use your Desktop to much better advantage.

Therefore, depending on what you actually do from day to day, you may choose to use only the VGA driver, or only the Windows/X driver. You may also choose to switch back and forth. Switching display drivers involves running **SETUP** as described above. However, once you have installed the X driver the first time, you no longer have to pick "Other" and give a directory name. The "DOS Merge Windows/X driver" will be displayed as an option on the list of available display drivers.

Using the Windows version of **SETUP** is convenient because you can switch back and forth between VGA and X drivers from within Windows itself.

The DOS (or text based) version of **SETUP** can be useful in situations where it might be hard to start Windows. One prime example is the situation where you "lock yourself out" of Windows by starting it with the wrong display driver selected.

The Windows/X display driver is specific to X Windows. It will not work properly outside the Desktop (X Windows) environment. If you configure Windows to use this driver, Windows produces a display only if started in the X Windows environment. You can start it from an X icon or from the ScoTerm command line, and it will display properly.

CAUTION If you start Windows outside the Desktop (X) environment, it fails. Depending on your setup, you may get only a blank screen, the Windows logo and then a blank screen, or a return to the DOS command line with no explanatory message. You cannot navigate the windows desktop to run Windows' own internal **SETUP** program. These symptoms are the same for the use of any unsupported display driver (EGA and XGA cards, for instance).

To correct this, switch to your Desktop and double-click WinSetup to start an appropriate DOS environment (if you are not already there). From the DOS command line, switch to the directory where Windows is installed and run the DOS-based version of **SETUP**.

Size of Windows under X

The DOS Services Windows/X driver automatically sets the size of the X window in which Windows runs. The size is based on the resolution of your server. Your Windows window will be about 80% of your display size. On a server with a resolution of 800x600 this yields a window that is about 640x480. On a server with a resolution of 1024x768 this yields a window that is about 800x600.

Windows performance under X

The performance of Windows under X is around 70% of the performance of Windows in a Zoomed VGA screen. However, you can approach and even better the performance of a Zoomed VGA screen if you use an accelerator video card that is supported by your X server.

Switching mouse drivers

DOS Services always sees the mouse as a bus-connected, two-button, Microsoft-compatible mouse. If your real hardware is different, DOS Services uses its input, emulates the workings of a Microsoft Bus mouse, and passes that input along to your programs. To run under DOS Services, all your DOS software must be configured to expect input from a Microsoft Bus Mouse. This includes Windows.

If you install Windows (or any DOS program) under "raw" DOS, you must configure it to take input from whatever hardware you actually have. When you then run the software under DOS Services you must reconfigure it to accept the emulated mouse input passed to it by DOS Services.

In the case of Windows and all its applications, this is accomplished in a single reconfiguration. The situation is similar to switching back and forth between the VGA and the X driver, and the solution is the same. In brief:

1. Start an appropriate DOS environment by double clicking the WinSetup icon.
2. Change to the Windows directory.
3. Run the DOS text-based version of **SETUP**.
4. Change your mouse option to "Microsoft Mouse." **SETUP** may ask you for your Windows disks if this is the first installation. After that "Microsoft Mouse" will be available in the list of options.
5. Exit **SETUP**. You can now run Windows under DOS Services, but you must switch back to the driver appropriate to your real hardware the next time you run Windows under "raw" DOS.

Mouse tracking under X

When you run Windows on your Desktop you must have your Windows Mouse Tracking speed set to the slowest possible value. The Widows/X driver does this for you automatically when you first install it, However, if you use the Windows Control Panel to change your mouse tracking speed to anything else, your mouse will not function correctly in a Windows window. If this happens, you must use appropriate keyboard input to access the Windows control Panel and run **SETUP**. Set the mouse tracking back to slow.

Alternatively, you can exit Windows, use the DOS version of **SETUP** to change your display to VGA, start Windows in VGA mode and use the mouse to reset the mouse tracking. You can then use **SETUP** again to change your display driver back to X.

Colors under X

The number of colors available in your Windows environment depends on the number of colors available on your X server. On a 16 color server Windows will have a very limited range of colors. On a 256 color server you will have a much more extensive range of colors. On a 2 color server Windows will run in black and white.

Windows builds its icons during the installation of its display driver. If you change your color settings or if you change your colormap with the ScoColor utility, you must rebuild your Windows icons. To force Windows into rebuilding its icons, use **SETUP** to change your display driver to VGA and then back to Windows/X.

Installing Windows applications

To install a Windows application, log into your UNIX system as yourself, start a Windows session, and follow the normal installation procedures for the application. Since many applications assume the root directory of C: for their default installation directory, you may have to specify an alternate directory.

Some applications need more memory than others. Refer to the application documentation before configuring specific Windows applications.

If the application does not allow you to specify a subdirectory, then specify the *D:* drive. Under DOS Services, *D:\* is automatically substituted to your home directory with the DOS **SUBST** command.

Do not allow an application's installation routine to alter your personal *AUTOEXEC.BAT* and *CONFIG.SYS* files. Perform those alterations yourself. Back up both of these files before any installation, just in case the installation routine makes changes without your permission.

After successfully installing a Windows Application, you may execute that application from the Windows desktop or directly from the UNIX system.

Chapter 20

Getting information about other computers

Each computer on a network has a unique name. Generally, each user within a site also has a unique user name. To communicate with computers and users across the network, you need to know the name of your computer and the names of other computers and users that you want to contact. Several simple network commands provide this information.

Finding computer names and user names

You can find out the name by which your computer is known on the network by entering the **hostname** command without any arguments. (A HOST is any computer on the network.)

To see a list of the names of other computers on your network, use the **ruptime** command. (The " r " in the command name stands for "remote.") A computer that you access across the network is called the REMOTE computer, while a computer that you log in to directly is the LOCAL computer.) The output from the **ruptime** command shows the amount of time each computer has been up on the network.

```
cicely    up    26+15:00,    1 user,    load 2.08, 2.01, 2.00
brick   down        4:07
rosalyn   up    43+01:27,    4 users,   load 3.00, 3.00, 3.00
```

The **ruptime** manual page explains the displayed status information.

Network

To find out a person's user name, use the **finger** command and give the person's first or last name. For example, for a person named Maggie O'Connell, you might enter:

finger maggie

The response might be something like this:

```
Login name: maggieo                     In real life: Maggie O'Connell
Directory: /u/maggieo                   Shell: /bin/sh
No Plan.
```

You now know that you can send mail to Maggie through her user name *maggieo.*

(Of course, you could have just asked Maggie what her user name is.)

Chapter 21

Logging in to another computer

Logging in to a computer over the network means that your terminal acts as if it were attached directly to that computer.

The commands **rlogin**, **telnet**, **ct**, **cu**, and **vtp** let you log in to another computer while you remain logged in on your computer. To choose the method that suits your needs, consider the following:

- **rlogin** (page 204) only works when you are logging in to a UNIX system connected to yours with TCP/IP (Transmission Control Protocol/Internet Protocol). **rlogin** can be very convenient, if either you or the system administrator for the other computer arranges for you to use it to log in without entering your user name and password.
- **telnet** (page 205) lets you connect to computers running a variety of operating systems, if they are connected to your computer with TCP/IP.
- **ct** (page 206) connects a computer to a terminal across the phone lines, using UUCP software.
- **cu** (page 207) connects one computer to another computer, usually across phone lines, using UUCP software. If you connect to another UNIX system, you can move back and forth between both computers.
- **vtp** (page 209) lets you log in across a LAN Manager network.

To find out what software your system is running, ask your system administrator.

Once you log in to a computer across the network, you can use any command that you would use when logged in directly, including screen-oriented programs like **vi**. (If you log into a computer that uses a different operating system, you must use that operating system's commands.) If you want to use icons and perform other graphics-oriented tasks on the remote computer, ask your system administrator whether the local X server and the environment on the remote X system are configured to do so.

Controlling access between computers

System administrators determine which users have network access to their computers. If you find you cannot run certain commands across the network, ask your system administrator whether you have access to your target computer.

For the commands **rlogin**, **rcmd**, and **rcp** (explained in this and the following chapters), you can control who has access over the network to your accounts by creating a *.rhosts* file in your home directory on each computer. If the system administrators have not established access between two computers on which you have accounts, you can use the *.rhosts* file to establish your own access between your accounts on the two computers. The **rhosts** manual page describes how to use this file.

Logging in with rlogin

To log in from your computer to a remote computer called *cicely*, use the **rlogin** (remote login) command with the name of the remote computer:

```
rlogin cicely
```

cicely must be running a UNIX operating system and be connected to your computer with TCP/IP. If your login name is different for the remote computer, see the **rlogin** manual page for instructions.

Normally, **rlogin** prompts you for a password. Enter the password for your account on *cicely*, and you are logged in. You can run commands on *cicely* just as you would on your computer.

If you do not want to enter your password each time, you can ask the system administrator on *cicely* to establish automatic login at the system level. If you have an account on *cicely*, you can set up automatic login for yourself by creating a *.rhosts* file in your home directory on *cicely* (see the **rhosts** manual page for more details).

Your system administrator can also also arrange for you to log in to another computer simply by entering the name of that computer on the command line (without specifying the **rlogin** command). For this to work, your system administrator must create a link in the */usr/hosts* directory for each computer you want to log in to, and you must have this directory in your search path.

Ending an rlogin session

When you are finished with your work on the other computer, log out to end the **rlogin** session. You return to the computer from which you started, and **rlogin** tells you that the remote connection has been closed. If, for some reason, you cannot end an **rlogin** session normally, type a tilde followed by a period (~.) on a line by itself. This action aborts the session on the other computer and returns you to your computer.

NOTE When you are logged in to another computer with **rlogin**, you cannot normally type a tilde at the beginning of a line because the tilde is the default escape character. If you need to type a line that begins with a tilde, you must type two tildes (~~).

You can change the **rlogin** escape character with the **-e** option (see the **rlogin** manual page for details). If you do so, you must type the new escape character twice when you want it to appear at the beginning of a line.

Logging in with telnet

telnet is more flexible than **rlogin** (in that it works with any operating system), but it is less convenient (in that you always have to enter a user name and password).

To log in from your computer to a computer called *cicely*, enter the **telnet** command, followed by the name of the other computer:

```
telnet cicely
```

telnet prompts you for the user name and password of your account on *cicely*. Enter them. When you see the shell prompt from *cicely*, you can enter commands as you would on your computer.

Using telnet commands

telnet also has its own command line, from which you can enter special **telnet** commands. The **telnet** command line prompt looks like this:

```
    telnet>
```

To reach the command prompt on your local computer, give the **telnet** command without a computer name. To reach the command prompt during a **telnet** login session on another computer, enter the **telnet** escape character ⟨Ctrl⟩] on a line by itself.

At the command prompt, enter any **telnet** command (enter **?** or see the **telnet** manual page for a list of commands with descriptions). With **telnet** commands, you can define how your computer communicates with other computers. You can also log into another computer from **telnet**'s command prompt with the **open** command:

```
    telnet> open cicely
```

The **status** command shows whether or not you are connected to another computer, lists the current option settings (if you are connected to another computer), and displays the current escape character.

You can abbreviate a command as long as you enter enough characters to distinguish it from other **telnet** commands.

Ending a telnet session

When you finish your work on the other computer, log out from the shell prompt to end the **telnet** session. You return to your local computer.

To disconnect at the **telnet** command prompt, either enter **close** to disconnect from the remote computer and return to your local **telnet** session, or enter **quit** to exit **telnet** completely.

Logging in with ct

The **ct** (call terminal) command uses phone lines to connect a computer to a terminal equipped with a modem and allows a user on that terminal to log in to the computer. The receiving modem must be able to answer the call automatically.

You can use **ct** from the terminal end when you want to avoid long distance charges. From the terminal, call the computer that you want to access, log in, and issue the **ct** command to have the computer place a call to your terminal. The computer hangs up the line and calls your terminal back.

For example, if you want to work from home over a 1200-baud modem and your home phone number is 932-3497, dial the computer at work, log in, and enter the following command:

```
ct -s1200 9323497
```

If **ct** cannot find an available dialer, it tells you that all dialers are busy and asks if you want to wait until one becomes available. If you answer **n** (no), **ct** quits. If you answer **y** (yes), it asks how long (in minutes) it should wait.

When **ct** finds an available dialer, you are asked whether you want to hang up. Because you want to avoid long-distance charges by having the computer call you, answer **y** to hang up. The **ct** program logs you off and then calls your terminal back.

See the **ct** manual page for a complete list of **ct** options.

Logging in with cu

The **cu** (call UNIX) command logs you into a computer that is connected to your computer through a phone line (with modems) or through a direct serial line (without modems). The other computer does not have to run a UNIX operating system. However, if the other computer is a UNIX system, you can move back and forth between the two computers, transferring files and executing commands on either.

NOTE You can only transfer ASCII text files with **cu**. For information on transferring binary files to another UNIX system, see "Sending files with uucp" (page 223).

For example, to log in to a UNIX system at phone number 847-7867, enter the following command:

```
cu -s1200 8477867
```

The **-s1200** option causes **cu** to use a 1200-baud dialer. If you do not specify an **-s** option, **cu** uses the first available dialer at the speed specified in the */usr/lib/uucp/Devices* file.

When the other UNIX system answers the call, **cu** prompts you for your login. Enter your login and password. You can now use this computer as if you were logged in locally.

You can specify the remote system by name, rather than phone number, if the system name appears in the local computer's */usr/lib/uucp/Systems* file. For example, to log in to the system *cicely*, enter the following command:

```
cu cicely
```

Do not specify a baud rate or phone number when you specify the system name, because **cu** gets that information from the *Systems* file.

See the **cu** manual page for a list of **cu** options.

Using cu commands

When you are logged in to another UNIX system with **cu**, **cu** recognizes several special command strings. For example, you can run a command on your local system by preceding the command with a tilde and an exclamation point (~!). So, to see your working directory on your local computer, you can enter the following:

```
~!pwd
```

In these command strings, the tilde must be the first character on a line. Therefore, it is a good idea to press ⟨Enter⟩ before using a command string.

Two more command strings are **~%take** and **~%put**. The **~%take** command lets you copy files from another computer to your computer, while the **put** command lets you copy files from your computer to another computer. For example, to copy a file named *proposal* from the current directory of the remote computer to your home directory on your local computer, enter the following command:

```
~%take proposal $HOME/proposal
```

To copy a file named *minutes* from your home directory on your local computer to a file named *minutes.9-18* in the */tmp* directory of the remote computer, enter the following command:

```
~%put $HOME/minutes /tmp/minutes.9-18
```

The **take** and **put** commands copy only text files, and only to UNIX systems. You cannot use these commands to copy binary files.

NOTE The **cu** command cannot detect or correct transmission errors. After a file transfer, you can check for loss of data by running the **sum** command on both the file that was sent and the file that was received. This command reports the total number of bytes in each file. If the totals match, your transfer was probably successful. See the **sum** manual page for details.

Ending a cu session

When you finish your work on the remote computer, log out to end the **cu** session. If, for some reason, you cannot log out of the other computer normally, enter a tilde followed by a period (~.) on a line by itself. This action aborts the **cu** session. You are still logged in to the machine from which you started.

Logging in with vtp

If your network is running LAN Manager software, you can use **vtp** (virtual terminal program) to log into a XENIX-NET or LAN Manager for UNIX Systems server. Ask your system administrator whether you can use **vtp**.

To log in to the computer *brick* using **vtp**, enter:

```
vtp brick
```

You are prompted for your login name, password, and terminal type. After you enter them, you are logged into *brick*.

Running commands locally with vtp

With **vtp**, you can execute a command on your local computer while logged into another computer. At the beginning of a line, enter a tilde and an exclamation point (~!), followed by the command.

For example, to list the contents of the current directory on your local computer, enter this command:

```
~!ls
```

After the command finishes, you are prompted to continue your **vtp** session.

Ending a vtp session

When you finish your editing work on *brick* and want to end the session, log out. You see *brick*'s login prompt. Enter a tilde followed immediately by a period (~.). The **vtp** connection terminates.

Chapter 22

Running commands on another computer

The commands **rcmd, uux, lp,** and **rprint** all let you execute commands on a remote computer without having to log in to that computer. To decide which command suits your need, consider the following:

- **rcmd** (page 212) only executes commands on other UNIX systems that are connected to yours with TCP/IP. You or the system administrator on the other computer must initially arrange for you to use **rcmd.**
- **uux** (page 212) works on computers connected to yours by UUCP across telephone lines. **uux** might wait several hours before transferring your file.
- **lp** (page 213) sends a print job across a TCP/IP network.
- **rprint** (page 213) sends a print job across a LAN Manager network.

By running commands on another computer, you can take advantage of idle CPU time on the other computer, share printers, hard disks, or other remote devices, run software that resides on another computer, or find out information about the other computer.

You cannot use the commands discussed in this chapter to run interactive programs, such as vi. To use an interactive program on another computer, first log in to that computer remotely (see Chapter 21 (page 203)).

Running commands with rcmd

With **rcmd** (remote command), you can send commands to another computer running a UNIX operating system and receive the results on your computer. Either the system administrator for the remote computer must initially arrange **rcmd** access for you; or, if you have an account on the remote computer, you can arrange **rcmd** access for yourself by creating a *.rhosts* file in your home directory on the remote computer (see the **rhosts** manual page for details.)

To do a long listing of the files in the directory */w/quarter1* on the remote computer *cicely*, enter the following command:

```
rcmd cicely l /w/quarter1
```

The **rcmd** command uses any aliases it finds in your shell startup file (*.cshrc* for C-shell users or *.profile* for Bourne shell users) in your home directory on the other computer.

When you want to use a special character (such as " < " or " > " for redirection, " | " for piping, or " & " for process control) as part of the remote command, precede the character with a double or single quotation mark (" or ') or a backslash (\). Any special character not prefaced by a quotation mark or backslash is interpreted on your local computer.

For example, to run a long listing of the files in the directory */w/quarter1* on the remote computer *cicely* and save the output to a file on *cicely* called */tmp/quarter1.list*, enter the following command:

```
rcmd cicely ls /w/quarter1 \> /tmp/quarter1.list
```

If you leave out the backslash before the redirection symbol, **rcmd** creates the file */tmp/quarter1.list* on your local computer, instead of on *cicely*.

Refer to the **csh** or **sh** manual page for complete information about using special characters on the command line.

Running commands with uux

Use **uux** (UNIX-to-UNIX execution) to run commands that affect files on computers connected to yours by UUCP. UUCP is a series of UNIX networking programs usually used over telephone lines. (See "Copying files with UUCP programs" (page 222) for more information on UUCP.)

For example, to send the file */tmp/printfile* from the remote computer *brick* to *brick*'s default printer, enter the following:

```
uux brick!lp brick!/tmp/printfile
```

Prefixing a command with the UUCP computer name causes the command to be executed on that computer.

If your computer is connected to two computers, *brick* and *rosalyn*, you can compare the contents of */tmp/chpt1* on *brick* with */tmp/chpt1* on *rosalyn* by entering the following command:

```
uux "diff brick!/tmp/chpt1 rosalyn!/tmp/chpt1 > diff.file"
```

This command compares the contents of the files and places the output in *diff.file* in the current directory on the local computer. Because there is no UUCP computer name prefixed to the **diff** command, the command runs locally.

In the example above, the **uux** command line is placed in quotation marks because it contains the redirect symbol "> ". Place the **uux** command line in quotation marks whenever the command line contains special shell characters such as <, >, |, and &.

Commands sent with **uux** are not sent immediately. Instead, they wait in a spool directory until the appropriate daemon (in this case, **uucico**) awakens. Depending on how your system is configured, a **uux** command might take place within minutes, or it might take hours.

For security reasons, the commands available under **uux** are often limited. If you attempt to execute an unavailable command, you receive mail indicating that the command cannot be executed on the remote computer.

See the **uux** manual page for options you can use with **uux**.

Printing a file on a remote printer

If your system is running TCP/IP software, you can use the **lp** (line printer) command to print a file on a remote printer, just as you print a file on a local network. To send a print job from your computer to a printer on the LAN Manager network, you can use the **rprint** (remote print) command. Ask your system administrator which command you should use.

Using lp, lpstat, and cancel

If the remote printer is your default printer, enter the following to print the file *proposal*:

```
lp proposal
```

To send the file *proposal* to the non-default printer *moose*, enter:

```
lp -dmoose proposal
```

You can also use the **-n** option to designate a certain number of copies and the **-t** option to print a title on the banner page of the printout. Options other than **-d**, **-n**, and **-t** will be ignored for remote printing.

To see the status of **lp** requests on the printer *moose*, enter:

```
lpstat -pmoose
```

You can also use the **-a** option to see the acceptance status of **lp** requests, the **-o** option to see the status of output requests, and the **-v** option to see the pathnames of devices associated with the printers. Options other than **-a**, **-o**, **-p**, and **-v** will be ignored for remote printing.

To cancel a print request that has not yet printed, use the **cancel** command with the job number from the **lpstat** display. For the print job "laserwriter-635," enter:

```
cancel laserwriter-635
```

See "Printing a file" (page 87) and the **cancel**, **lp**, and **lpstat** manual pages for more information on these commands.

Using rprint

If your system is running LAN Manager, use **rprint** to send a print job to a printer across the network.

Before you use **rprint**, you need to connect to the computer that controls the remote printer.

Connecting to a remote printer

To connect to the machine that controls a remote printer, use the **net use** command. The exact syntax for this command depends on the software running on the remote computer. It usually includes an alias, the name of the remote computer, the name of a printer or a user account on the remote computer, and a password for the printer or the user account.

The alias is a short name of your choice. You use it within the **rprint** command to refer to the printer or user account. Do not choose an alias name that you are using for another LAN Manager connection. Also, do not use the name of another node on the network as the alias. To find out the aliases of your current connections, enter the **net use** command (with no arguments) or the **net stat client** command. Ask your network administrator for a list of network node names to avoid.

Ask your system administrator for the names and passwords for printers and user accounts to which you have access. If you omit the password from the command line, you are prompted to enter it. This is a good method for keeping the password secure, because it does not appear on the screen when you type it.

In the following example of **net use**, you connect to the shared printer *moose* on the computer *brick*:

```
net use moo //brick/moose prntpswd
```

In this command line, *moo* is your alias for the printer *moose*, and *prntpswd* is the printer's password.

See "Sharing files under LAN Manager" (page 251) for a full explanation of the **net use** command.

Sending a file to a LAN Manager printer

Once you connect to a remote printer with the **net use** command, send the file *report* by entering a command like the following:

```
rprint moo report
```

moo is the alias for the remote printer.

If you do not specify a filename in the **rprint** command, the command prints from the standard input, such as the keyboard or a pipeline.

You can use **rprint** from an application program, such as a word processor with a configurable print option. Specify the full **rprint** command line as the print command to be used from your application.

The **rprint** manual page describes more options for **rprint**.

Chapter 23

Copying files between computers

You can use the commands **rcp, ftp, uucp,** and **uuto** to transfer files between computers on the network. To decide which command suits your need, consider the following:

- **rcp** (page 218) only copies files between UNIX systems running TCP/IP. However, you can copy an entire directory at once.
- **ftp** (page 219) copies ASCII or binary files between UNIX or non-UNIX systems running TCP/IP. With anonymous **ftp,** you do not need an account on the target computer. **ftp** syntax can be complicated.
- **uucp** (page 222) transfers a file to or from a UNIX system connected to yours by UUCP, usually across telephone lines. UUCP might wait several hours before transferring your file. With **uucp,** you can specify which directory on the target system receives the file. The **uucp** syntax can be complicated.
- **uuto** (page 222) also transfers a file to or from a UNIX system connected to yours by UUCP, usually across telephone lines. However, **uuto** restricts where, on the target computer, you can place the file. UUCP transfers can take several hours.

Copying files between DOS, OS/2, UNIX systems

When you copy or share files between UNIX systems and DOS or OS/2 systems, be aware that file formatting conventions differ between the systems. The **dtox** and **xtod** commands convert files from one format to the other. (**dtox** stands for DOS-to-UNIX, and **xtod** stands for UNIX-to-DOS.) Not all programs are sensitive to file-format differences; generally, you only need to change text and data files that must be in strict format. You do not need to convert binary object files.

The following command converts the file *report.dos* from DOS or OS/2 format and saves the UNIX system format file in *report.unx*:

```
dtox report.dos > report.unx
```

The following command converts the file *memo.unx* from UNIX system format and saves the DOS or OS/2 format file in *memo.dos*:

```
xtod memo.unx > memo.dos
```

Copying files with rcp

You can use **rcp** (remote copy) to copy files to or from a UNIX system that is connected to your computer with TCP/IP. Either the system administrator for the remote computer must initially arrange your access through **rcp**; or, if you have accounts on both machines, you can arrange **rcp** access between your two accounts by creating a *.rhosts* file in your home directory on the remote computer. (See the **rhosts** manual page for details.) If you often work with another UNIX system, **rcp** is convenient because it does not require you to enter your user name and password for the other computer, and it allows you to copy an entire directory at once.

The syntax of **rcp** is similar to the UNIX **cp** command; but, with **rcp**, you can precede either the source path or the destination path with a remote computer name. You separate the computer name from the filename with a colon. The square brackets in the following **rcp** syntax description indicate that the computer name is optional for both the source and destination files:

rcp [*computer*:]*directory.../filename* [*computer*:]*directory.../filename*

Copying your own files with rcp

To copy the file *proposal* from the directory */u/proj3/design* on the remote computer *cicely* to the current directory on your computer, enter:

```
rcp cicely:/u/proj3/design/proposal proposal
```

The user *ruthanne* can copy her weekly report from her personal *status* directory on her local computer to the group *status* directory on the remote computer *cicely* with this command:

```
rcp /u/ruthanne/status/engr.09.29 cicely:/u/staff/status
```

To copy a directory, use the **-r** option. For example, to copy the */u/proj3* directory (with all its subdirectories and files) from the remote computer *cicely* to a subdirectory named *proj3* in your current directory on your computer, enter:

```
rcp -r cicely:/u/proj3 proj3
```

With the **-r** option, the destination must be a directory.

By specifying a computer name for both the source and destination files, you can copy a file between two remote computers without first moving the file to your local computer. From your local computer, you can use the following command to copy the file *notes* from your home directory on *cicely* to your home directory on the computer *rosalyn*:

```
rcp cicely:notes rosalyn:notes
```

Copying other users' files with rcp

You can use the **rcp** command only to access files and directories to which you would ordinarily have access according to UNIX file permissions. **rcp** verifies file access permissions with the user name under which you are logged in on your computer. Another user, for example, *maggieo*, on a remote computer can give you access to her account by creating a *.rhosts* file in her home directory (see the **rhosts** manual page for details). Then, you can copy *maggieo*'s files by specifying her user name in the **rcp** command line. For example, if *maggieo* is on *cicely*, you can copy the file *personal/letter* from *maggieo*'s directory with:

```
rcp maggieo@cicely:personal/letter letter.maggieo
```

Because the pathname for the file *letter* is not an absolute path, **rcp** assumes the path is relative to the specified user's home directory.

Copying files with ftp

ftp (file-transfer program) is an interactive program with its own set of commands for connecting to a remote computer and accessing files. With **ftp**, you can copy ASCII or binary files to a computer running either a UNIX or non-UNIX operating system. The remote computer must be connected to yours with TCP/IP. The **ftp** command also offers certain file-transfer privileges for users who do not have an account on the target computer. To attain this flexibility, **ftp** uses a more complex syntax than **rcp**.

Connecting to another computer with ftp

To connect to a remote computer, enter the **ftp** command followed by the computer name. Then enter your user name and password when prompted. When the connection is made, you see the **ftp** prompt:

```
ftp>
```

At the **ftp** prompt, you can give any **ftp** command. Enter **?** for a list of available commands (see the ftp manual page for descriptions). When your **ftp** command finishes processing, **ftp** displays its prompt again. You remain in **ftp** command mode until you exit **ftp** with the **quit** command.

NOTE When you use this version of **ftp** under a given account name, the shell for that account must appear in */etc/shells*, or you cannot start **ftp**.

You can also reach the **ftp** prompt on your local machine, without connecting to another computer, by entering **ftp** without specifying a computer name. You can then use the **open** command to connect to another computer. For example, to connect to the computer *brick* from the **ftp** prompt on your own computer, enter this command:

```
ftp> open brick
```

By default, **ftp** operates in verbose mode, displaying many messages about how it performs your file-transfer requests. To turn off verbose mode, enter the **verbose** command at the **ftp** prompt. (The **ftp** examples in this chapter were created with verbose mode turned off.)

Setting up automatic ftp connection

You can eliminate the need to enter your user name and password for each **ftp** connection by setting up a *.netrc* file in your home directory on your computer.

In the *.netrc* file, put your user name and, optionally, your password for each computer to which you want **ftp** to connect automatically. When you try to connect to a particular computer, **ftp** searches *.netrc* for an entry for that computer. If login information exists, **ftp** automatically supplies the information to the other computer. Because *.netrc* contains passwords in unencrypted text, it creates a security risk you might want to avoid. In that case, omit the password from each entry in *.netrc*. **ftp** then prompts you for it when you connect.

If you do include your password in *.netrc*, **ftp** requires that you set the permissions on the file to 400 or 600, so that other users cannot read it. If the file contains your password and has the wrong permissions, ftp aborts the login process.

See the **netrc** manual page for details about this file. See the **chmod** manual page for more information on file permissions.

Using get and put

After you have connected to another computer, you can copy a file from that computer to your computer with the **get** command. Give the pathname of the file you want to transfer from the other computer, followed by the destination on your computer. For example, enter the following to copy the file */usr/local/lib/bugreport/bugs.form* from the remote computer *brick* to your current directory on your local computer:

```
ftp> get /usr/local/lib/bugreport/bugs.form bugs.form.new
```

Use the **put** command to copy a file from your computer to a remote computer. For example, after you have edited the bug report form, use **put** to copy your new version to *brick*:

```
    ftp> put bugs.form.new /usr/local/lib/bugreport
```

Copying files to or from a non-UNIX system

Different types of operating systems use different file formats. When **ftp** transfers files to or from a non-UNIX system, it transfers them in ASCII mode by default and automatically translates from one system's format to another when appropriate. (You do not need to use **dtox** or **xtod** to convert the files.) Use this default ASCII mode when transferring ASCII text files with a non-UNIX system.

If you want to transfer binary files with a non-UNIX system, enter the **binary** command at your **ftp** prompt. To switch back to ASCII mode, enter the **ascii** command. In general, if you transfer a file and get unexpected results, try again in the other mode.

When transferring files between 386 UNIX systems, both modes work properly for both ASCII and binary files because no translation is needed. Binary mode works faster.

Using anonymous ftp

If the system administrator on another computer has set up a public **ftp** account, you can transfer files within certain protected directories of the **ftp** home directory without an account on that computer. Although a public **ftp** account has restricted file access, it can be very useful for exchanging public software or other information.

To use this public account, make your **ftp** connection with the user name *anonymous*. This account has no password, but it is customary to enter your user name at the password prompt.

An error message saying that the *anonymous* user is unknown usually means that the system administrator of the other computer is not allowing public access.

Ending an ftp session

When you finish your work across an ftp connection, either enter the **close** command to disconnect from the other computer and return to the **ftp** session on your machine, or end the **ftp** session completely with the **quit** command.

Copying files with UUCP programs

UUCP is a series of programs that provide networking capabilities for UNIX systems. While UUCP commands can be used over direct serial lines, they are usually used on computers connected by telephone lines.

With the UUCP programs **uucp** and **uuto**, you can transfer files to and from other computers. When the computers are connected by telephone lines, UUCP transfers can take place across thousands of miles to any other UUCP site in the world.

Both **uucp** and **uuto** can be used to transfer copies of binary and text files between UUCP sites. There are advantages and disadvantages to each. The **uucp** command gives you great flexibility in specifying where on the other system the transferred file is to be placed. However, **uucp** syntax can be rather long and complicated. In contrast, the **uuto** command is easy to use, but it restricts where you can place the file on the other system. In addition, retrieving a file sent with **uuto** is slightly more complicated than retrieving a file sent with **uucp**.

Before you try to use **uucp** or **uuto**:

- Verify with your system administrator that your site can dial out to other sites.
- Make sure that your site can reach the other site. Enter the **uuname** command with no options for a list of UUCP sites to which your computer talks directly.

  You might be able to communicate with a site that does not show up in a **uuname** listing by naming more than one site name on the command line. See "Sending files through intermediate sites" (page 224) for more information.
- Make sure that the file you send has read permission set for others and that the directory containing the file has read and execute permissions set for others. Use the **l** (lowercase "L") command to check the file's permissions and the **l -d** command to check the directory's permissions. If the permissions are not correct, enter the following commands to set the correct permissions:

  ```
  chmod o+r filename
  chmod o+rx directory
  ```

Sending files with uucp

With the **uucp** command, you can copy a file to any directory on the remote computer, *if* your computer has write permission in the chosen directory. For example, to send the file */usr/mauricem/proposal* from your local computer to the */usr/maggieo* directory on the computer *brick*, enter the following command:

```
uucp /usr/mauricem/proposal brick!/usr/maggieo/proposal
```

NOTE If you are working in the C-shell, you must precede any exclamation marks with a backslash (\). For a C-shell user, the command above is entered as:

```
uucp /usr/mauricem/proposal brick\!/usr/maggieo/proposal
```

You can replace the pathname of a user's home directory with ***~username.*** Also, you do not need to supply the full pathname to designate a file in your current directory. For example, if */usr/mauricem* is your current directory, and */usr/maggieo* is *maggieo*'s home directory on the computer *brick*, enter the following command to copy */usr/mauricem/proposal* from your local machine to the directory */usr/maggieo* on *brick*:

```
uucp proposal brick!~maggieo/proposal
```

The only way to find out which directories your computer can write to on the other site is to contact somebody at the site. By default, most UUCP sites permit calling-in computers to write to their */usr/spool/uucppublic* directory, so this is the safest directory to target with a UUCP transfer. You can replace */usr/spool/uucppublic* in the command line with a tilde (~). For example, to send the file */usr/mauricem/proposal* from your local computer to the */usr/spool/uucppublic* directory on the computer *brick*, enter the following command:

```
uucp /usr/mauricem/proposal brick!~/proposal
```

This command creates the file */usr/spool/uucppublic/proposal* on *brick*.

You can also use the **uucp** command to obtain files from another computer. For example, if you want to copy */usr/maggieo/proposal* from the remote computer *brick* to the */usr/mauricem* directory on your computer, enter the following command:

```
uucp brick!/usr/maggieo/proposal /usr/mauricem/proposal
```

With the **uucp** command, files are not copied and sent immediately. Instead, copies are placed in a spool directory and sent once the **uucico** daemon awakens. Depending on how your system is configured, a **uucp** transfer might take place within minutes, or it might take hours.

Sending files through intermediate sites

You might be able to send files to a UUCP site not listed in a **uuname** listing. For example, if your computer is connected to a UUCP site named *brick,* and *brick* is connected to a UUCP site named *rosalyn,* you can send */tmp/proposal* on your computer to */usr/spool/uucppublic* on *rosalyn.* To do this, specify the full UUCP address relative to your computer:

```
uucp /tmp/proposal brick!rosalyn!~/proposal
```

Note that each site name in the command line is followed by an exclamation mark. By placing several site names in a **uucp** command line, you can greatly extend the range of systems to which you can copy files with **uucp**. This is also true for the **uuto** command discussed in "Sending files with uuto" (this page).

Monitoring your uucp transfers

Two useful options to **uucp** are **-m** and **-n** *user*.

The **-m** option sends you mail reporting on the success or failure of the file transfer. The **-n** *user* option notifies the *user* on the computer to whom the files are sent of the file transfer.

See the **uucp** manual page for a complete list of **uucp** options.

You can also use the **uustat** command to check on the status of files you copied with **uucp**. To check on the status of all your **uucp** jobs, enter the following command:

```
uustat
```

The output displays the job ID of the **uucp** request, the date and time, an " S " for a send job or an " R " for a request, the login name of the user who queued the job, the size of the file, and the filename.

See the **uustat** manual page for a complete list of **uustat** options.

Sending files with uuto

The **uuto** command lets you copy files to the public directory of a UUCP site to which your system is connected. The public directory on most XENIX and UNIX systems is */usr/spool/uucppublic,* and files sent with **uuto** are placed in the directory:

/usr/spool/uucppublic/receive/username/source_computer

Here, *username* is the login of the user to whom you are sending files and *source_computer* is the site name of your system.

For example, to send a copy of */tmp/proposal* from your computer, *cicely*, to *maggieo* on *brick*,

uuto /tmp/proposal brick!maggieo

This command copies *proposal* to the following directory:

usr/spool/uucppublic/receive/maggieo/cicely

When the file transfer is complete, a **mail** message notifies *maggieo* that the file has arrived. If you use the **uuto -m** option when you send the file, **mail** also notifies you of the success or failure of the transfer.

As with **uucp**, files transferred with **uuto** are not transferred immediately after the command is entered. Instead, they are placed in a spool directory and sent when the **uucico** daemon awakens.

Retrieving files with uupick

To retrieve a file sent by **uuto**, enter the **uupick** command with no arguments. The **uupick** program searches the public directory for any files sent to you. If it finds any, it prompts you to retrieve them. For example, if you successfully transfer the file *tmp/proposal* from *cicely* to *maggieo* on *brick*, *maggieo* sees the following **uupick** prompt:

```
from cicely: file proposal ?
```

To move the file *proposal* to her home directory, *maggieo* can enter the following in response to the **uupick** prompt:

m $HOME

maggieo could specify a different directory in place of **$HOME**. If *maggieo* does not specify a directory after **m**, the file is moved to her current directory.

To delete the file *proposal* from the public directory, *maggieo* can enter **d** at the **uupick** prompt.

Quit **uupick** by entering **q**. See the **uupick** manual page for a complete list of **uupick** options.

Chapter 24
Mailing files and messages

There are several mail programs that you can use to compose, send, receive, forward, and reply to electronic mail. You will probably find it most convenient to use the Mail accessory on the Desktop, which is described in Chapter 5 (page 49).

You can also use the UNIX command line **mail** utility. This chapter describes how to send mail (this page), read mail (page 231), customize your mail environment (page 237), and use some advanced mail features, such as automatic forwarding (page 240). Several tables at the end of the chapter summarize **mail** comands and options (page 245).

Sending mail

To start composing a message, enter the following command:

mail ***username***

Here, ***username*** is any user's login name, including your own.

If the **asksub** option is set, **mail** prompts for a "Subject:" line (see "Customizing your mail environment" (page 237). Enter a one-line summary of the message. After you press ⟨Enter⟩, begin composing the body of the message. While composing, you can use line-editing functions, such as ⟨Ctrl⟩U to kill a line and backspace to back up one character. End each line by pressing ⟨Enter⟩.

To view the message you are composing (including the heading fields) as it will appear when you send it, enter the following on a line by itself:

~p

You can abort a message while you are composing it by pressing ⟨Del⟩ (INTERRUPT) twice, or by entering **~q** on a line by itself. When you abort a message, a copy of the body of the undelivered message is saved to the file *dead.letter* in your home directory.

When you are ready to send your message, press ⟨Ctrl⟩**D** on a line by itself to end the message and send it. Once you have sent a message, there is no way to recall it, so be careful.

If mail cannot be delivered to the address you specified, you are notified with a message that includes the undeliverable message.

Sending mail to other sites

You can send mail to users on other computer sites that are networked to your site. For example, to send mail to the user *joelf* at the computer *cicely*, use one of the following mail-address formats:

```
mail cicely!joelf
mail joelf@cicely
```

Which format you use depends on the software running at the sites. Many sites use both formats. Ask your system administrator or the user at the other site which format you should use.

The address format shown in the first example (***sitename!username***) sends mail to UUCP sites connected to your site. If you use this address format, find out which UUCP sites your computer communicates with by entering the following command at the UNIX prompt:

```
uuname
```

A list of site names is displayed.

With this first address format, you might need to enter several site names on a command line, following each one with an exclamation point. For example, suppose that your site talks to UUCP site *cicely* and *cicely* talks to UUCP site *brick*. Send mail to user *ruthanne* on *brick* with the following command:

```
mail cicely!brick!ruthanne
```

NOTE If you are using the C-shell, you must precede each exclamation point with the backslash (\). A C-shell user enters the command above as follows:

```
mail cicely\!brick\!ruthanne
```

The address format shown in the second example (***username@sitename***) is the Internet format. You do not need to name intermediate site names in this format.

Executing shell commands from within mail

To execute a shell command while you are composing a **mail** message, precede the command with "~!". For example, to display the current date without leaving **mail**, enter:

~!date

To execute a shell command while reading the message header list, precede the command with an exclamation point.

To enter a new shell from the message header list, enter the **shell (sh)** command. To exit from this new shell and return to **mail** prompt, press ⟨Ctrl⟩D.

Editing mail headers

After you begin composing a message, you can add names to the list of message recipients with the following command:

~t ***login1 login2*** ...

You can name as many additional recipients as you like. You cannot remove anyone from the recipient list with **~t**; To remove a recipient, use the **~h** escape discussed later in this section.

You can replace or add a "Subject:" field by using the **~s** escape:

~s ***line_of_text***

This replaces any previous subject with *line_of_text*. The subject, if given, appears near the top of the message, prefixed with the heading "Subject:". You can see what the message looks like by using **~p**; this displays all heading fields along with the body of the text.

You might prefer to list certain people as recipients of carbon copies of a message rather than direct recipients. The following escape adds the named people to the "Cc:" list:

~c ***login1 login2*** ...

Similarly, the following escape adds the named people to the "Bcc:" (Blind carbon copy) list.

~b ***login1 login2*** ...

The people on this list receive a copy of the message, but are not mentioned in the message header.

The recipients of the message are given in the "To:" field; the subject is given in the "Subject:" field, and carbon copy recipients are given in the "Cc:" field. If you want to edit these fields in ways not possible with the **~t**, **~s**, and **~c** escapes, you can use:

~h

where " h " stands for "header." The escape **~h** displays "To:" followed by the current list of recipients and leaves the cursor at the end of the line. If you enter ordinary characters, they are appended to the end of the current list of recipients. You can also erase existing heading text by backspacing over it.

When you press ⟨Enter⟩, **mail** advances to the "Subject:" field, where the same rules apply. Another ⟨Enter⟩ brings you to the "Cc:" field, and another brings you to the "Bcc:" field. Edit each of these fields the same way. Finally, another ⟨Enter⟩ leaves you appending text to the end of your message body. Remember that you can always use **~p** to see what the message looks like.

Using a text editor in mail

To gain the full capabilities of a text editor while you are composing a message, enter one of these commands on a line by itself:

~e (to use **ed**, the line editor)

or

~v (to use **vi**, the visual editor)

When you finish editing the mail message with your chosen editor, write the message and quit the editor. mail responds with:

```
(continue)
```

You can finish composing your message, and then send it with ⟨Ctrl⟩**D**.

You can edit an existing message by entering either **ed** or **vi** from the message header list. For example, to edit message number 4, enter the following at the mail prompt " & ":

vi 4

When you finish editing the mail message, write the message and quit the editor. The edited message returns to its place in your mailbox, and you return to the mail prompt.

Including a file in a mail message

To send a message to *joelf* that consists solely of the pre-existing file *report*, you do not have to enter **mail** at all. Simply enter:

mail joelf < report

mail gives an error message if the file does not exist or cannot be read. To include the contents of the pre-existing file *report* inside a larger message, first begin composing the body of your message, and then enter the following on a line by itself:

~r report

If the read is successful, **mail** displays the number of lines and characters appended to your message. Then you can continue composing your message, and end with ⟨Ctrl⟩D.

Enter the following on a line by itself to read in the file *dead.letter* from your home directory:

~d

Reading your mail

Messages addressed to your user name arrive in your system mailbox, which is the file */usr/spool/mail/username*. To read your **mail**, enter:

mail

mail then displays a sign-on message and a list of message headers:

```
SCO Mail version 4.1  Type ? for help.
"/usr/spool/mail/username1": 2 messages
> N  1 username2    Fri Aug 31 12:26  9/229 "Sample Message"
  N  2 username3    Fri Aug 31  2:06  4/96  "Disregard"
&█
```

The ampersand prompts you to enter a **mail** command. For a complete list of **mail** commands, see the **mail** manual page. To get quick help on **mail** commands, enter a question mark at the ampersand prompt.

To display the first new message, press ⟨Enter⟩. To display any other message, enter the message number, then press ⟨Enter⟩.

Message headers are displayed a window at a time. You can set the size of the "screen" inside your UNIX window with the **screen** option (see Table 24-3 (page 246)). You can move forward one window with the **h** (**headers**) command:

```
h +
```

You can move backward one window with **h -**. Both plus and minus take an optional numeric argument that indicates the number of header windows to move forward or backward. With no argument at all, the **headers** command displays a window of headers in which the header of the current message is at the center.

Specifying message lists

Many **mail** commands take a list of messages as an argument. A message list is a list of message identifiers, ranges, users, search strings, or message types separated by spaces or tabs. For commands that take a message list as an argument, if no message list is given, the current message is used.

Message identifiers can be either decimal numbers, which directly specify messages, or one of three special characters: ^ (caret), . (dot), and $ (dollar sign), which specify the first, current, and last *non-deleted* messages, respectively.

A range of messages is two message identifiers separated by a dash. To display the headers of all the messages from the current message to the last message, enter:

```
h .-$
```

By giving a user name as part of a message list, you can display the messages sent by a particular user. For example, if you want to read only the messages sent by *maggieo*, enter:

```
p maggieo
```

The **p** command displays those messages one after another.

You can use a search string to specify all messages with the given string in the "Subject:" line (case is ignored). For example, to display the headers of only the messages with "meeting" in the "Subject:" line, enter:

```
h /meeting
```

You can create a message list by defining the type of messages in which you are interested. Use a colon followed by one of the following key letters:

d	deleted messages
n	new messages
o	old messages
r	read messages
u	unread messages

For example, to see a list of headers of the messages you deleted, enter:

```
h :d
```

Message lists can contain combinations of numbers, ranges, and names. For example, to delete all messages about your print jobs from user *lp* that are numbered from the first non-deleted message to 7 or 11 and 12, use the **d** command with the following message list:

```
d lp ^-7 11 12
```

In place of other message identifiers, you can use an asterisk (*) to mean all non-deleted messages. For example, to completely clean out your mailbox, use the **s** command with an asterisk and a filename to save all non-deleted messages to the specified file:

```
s * mail.old
```

Do not combine the asterisk symbol with any other message identifiers.

Replying to a message

Use the **r** (**reply**) command to respond to the author of a message right away. The **r** command automatically addresses the reply, and copies the original message's "Subject" field as the reply's "Subject". Compose a reply just as you compose an original message, and send it by pressing ⟨Ctrl⟩**D** on a line by itself.

The **R** (**Reply**) command works just like **r**, except that copies of the reply are also sent to everyone shown in the original message's "To" and "Cc" fields.

Including one mail message in another

If you are composing a message from within **mail**, you can insert a message that you have received into the message that you are composing. For example, enter:

```
~m 4
```

This reads message 4 into the message you are composing, shifted right one tab stop. Use the following escape to perform the same function, but with no right shift:

```
~M 4
```

You can name any non-deleted message or list of messages.

Saving a message

If you leave **mail** with the **q (quit)** command without performing any other operations on your message, your already-read messages are saved by default in your user mailbox (the file */u/username/mbox*).

To keep the message in the system mailbox (*/usr/spool/mail/username*) after you read it, use the **ho (hold)** or **pr (preserve)** command at the mail prompt. The following command holds message number 3 in the system mailbox:.

```
ho 3
```

If you do not specify a message number, you hold the current message.

Saving many messages in the user or system mailbox can be confusing and can slow down processing. You can use the **s (save)**, **w (write)**, or **W (Write)** commands to organize your mail by putting all related messages in a specific file (sometimes called a folder).

For example, the following command appends the current message to the file *letters*:

```
s letters
```

If the file *letters* does not already exist, **save** creates it. The **save** command writes the entire message, including the header fields (such as "To:", "Subject:", and "Cc:") and the ⟨Ctrl⟩**A** delimiters between messages to the specified filename.

Each saved message is marked with an asterisk (*). When you quit from **mail**, saved messages are normally deleted from the system mailbox.

You can access messages saved in a file by specifying the filename with the **mail -f** command-line option or with the **fold (folder)** or **fi (file)** command from within **mail**. Both of these methods give you access to the messages in that file in the same way you have access to the messages in your system mailbox when you invoke **mail** normally. You can access messages in */u/username/mbox* in the same way.

The **W (Write)** command works like **s**, but it saves your message text and headers without the ⟨Ctrl⟩**A** delimiters, so you cannot access the messages using **mail -f**. As with **s**, **W** marks the message with an asterisk, and the message is deleted from your system mailbox when you leave **mail**.

The **w (write)** command saves only the message text, without the headers or the ⟨Ctrl⟩**A** delimiters, to a plain text file.

Printing a message

You can send a message to the lineprinter with the **l (lpr)** command. This command takes a message list as its argument, then paginates and prints each message on the lineprinter. Each page of the printout is numbered and dated in the header. For example:

```
l ruthanne
```

prints each message from the user *ruthanne*.

Forwarding a message

Use the **f (forward)** command to send a copy of the current message to specified users. For example, to forward the current message to *mauricem*, enter:

```
f mauricem
```

The user *mauricem* receives the forwarded message, along with a heading showing that you forwarded it. The forwarded message is indented one tab stop inside the new message.

The **F (Forward)** command is identical to the lowercase **forward** command, except that the forwarded message is not indented.

Deleting a message

Unless you indicate otherwise, each message you receive is automatically saved in the user mailbox when you quit **mail**. Often, however, you do not want to save messages you have received. To delete a message, use the **d (delete)** command. For example:

```
d1
```

prevents **mail** from retaining message 1 in the user mailbox. The message disappears altogether, along with its number.

The **dp** command deletes the current message and displays the next message; this command is useful for quickly reading and disposing of mail.

The **u** (**undelete**) command causes a message that has been previously deleted with **d** or **dp** to reappear. For example, to undelete message 1, enter:

```
u1
```

You cannot undelete messages from previous **mail** sessions; they are permanently deleted.

Quitting mail

When you finish reading all your messages, you can leave **mail** by entering the **q (quit)** command. All messages are held in your user mailbox, except the following:

- deleted messages, which are removed irretrievably
- messages marked with the **ho** or **pr** command (these messages are saved in your system mailbox)
- messages saved with the **s** or **w** commands

Forwarded messages are *not* removed from the system mailbox. If the **hold** option is set (see Table 24-3 (page 246)), messages that you have read are saved in your system mailbox, not in your user mailbox.

To leave **mail** quickly without either deleting messages from your system mailbox or saving messages to your user mailbox, use the **x (exit)** command.

Customizing your mail environment

You can define your **mail** environment with switch and string options that can be set with the **mail** commands **set** and **unset**. A switch option is either on or off (set or unset). String options are strings of characters that are assigned values with the syntax *option=string*. Multiple options can be specified on a line. For example, in the following **set** command:

set dot askcc SHELL=/usr/bin/sh

the options **dot** and **askcc** are switch options; **SHELL** is a string option.

The **set** command with no arguments displays the options currently set.

You can create a personal mailing list with the **a** (**alias**) command. By using an alias, you can send mail to one name and have it go to a group of people. With no arguments, **alias** displays all currently defined aliases. With one argument, it displays the users defined by the given alias.

When you set **mail** options at the **mail** prompt (that is, from within the mail-header list), the settings you specify only remain effective during the current mail session. It is more useful to place **set**, **unset**, and **alias** commands in the file *.mailrc* in your home directory. The *.mailrc* file defines your personal default environment when you invoke **mail**. Whenever **mail** is invoked, it first reads the file */usr/lib/mail/mailrc*, then the file *.mailrc* in your home directory. System-wide **set** options and system-wide aliases are defined in */usr/lib/mail/mailrc*. These are installed by the system administrator on your system. Personal aliases and personal **set** options are defined in *.mailrc*.

The following is a sample *.mailrc* file:

```
# number sign introduces comments
# personal aliases office and cohorts are defined below
alias office bill joe sue
alias cohorts john mary bob beth mike
# set dot lets messages be terminated by period on new line
# set asksub prompts for Subject: before entering compose mode
set dot asksub
# cd causes mail to always begin executing from the same directory
cd
```

The following sections demonstrate how to create mailing lists and describe a few of the common **set** options. Refer to the **mail** manual page for details about other options.

Creating a mailing list

The **alias** command links a group of names with the single name given by the first argument, thus creating a mailing list. For example, you can create an alias called *beatles* with the following command:

alias beatles john paul george ringo

Now, whenever you used the name *beatles* in a destination address (you enter **mail beatles**), **mail** expands it to the four names aliased to *beatles.*

Aliases are expanded in mail sent to others so that the others can reply to each individual recipient. For example, the "To:" field in a message sent to *beatles* reads:

```
To:  john paul george ringo
```

and not:

```
To:  beatles
```

Keeping mail in the system mailbox

The **hold** option determines whether messages remain in the system mailbox when you exit **mail**. If you do not set **hold**, the examined messages are automatically placed in the *mbox* file in your home directory (your user mailbox). They are removed from the system mailbox when you quit **mail**.

Handling large amounts of mail

An effective way to organize your mail is to save messages in files (folders) organized by sender, by topic, or by a combination of the two. (Using mail folders is described in "Saving a message" (page 234).) However, this approach quickly eats up disk space. If any mailbox file (including your user mailbox) becomes large, examine its contents to decide whether messages are still relevant. To save space, consider summarizing very long messages.

You can create a directory to hold your mailbox files and define that directory to **mail** with the **folder** option. Then, whenever you save a message without giving a pathname, **mail** puts the message in a file in that directory. For example, if you want to save your messages by default in the directory *mail* in your home directory, use:

set folder=mail

If you forget the names of your mailbox files you can use the **folders** command to display the names of the files in the directory set by the **folder** option.

Sending carbon copies (CC)

The **askcc** switch causes prompting for additional carbon copy recipients when you finish composing a message. Responding with a ⟨Enter⟩ signals your satisfaction with the current list. Pressing INTERRUPT displays:

```
interrupt
(continue)
█
```

so that you can return to edit your message.

Listing messages in chronological order

The **chron** switch causes **mail** to list and display messages in chronological order. By default, messages are listed and displayed with the most recent first. Set **chron** when you want to read a series of messages in the order you received them.

The **mchron** switch, like **chron**, lets you read messages in chronological order, but lists them in the opposite order, that is, highest-numbered, or most recent, first. This is useful if you keep a large number of messages in your mailbox and you want to list the headers of the most recently received mail first but read the messages themselves in chronological order.

Converting mailboxes to the new MMDF format

If you are converting to SCO UNIX System V/386 from XENIX System V or another UNIX system, your old folders are in the older UNIX-system format. XENIX format uses the "From<space>" lines to to delimit between messages in mail folders; MMDF uses ⟨Ctrl⟩A characters.

By default, the **mail** program maintains the current format of every mail folder that you read with this program. For example, if a mail folder is in MMDF format (with ⟨Ctrl⟩A delimiters), **mail** uses this format when adding messages to the folder. If you use **mail** to read a folder that has not been converted to MMDF format, it prompts you to convert the folder.

If you use a Mail User Agent (MUA) other than **mail**, you should convert your old folders to use the new MMDF format using the **cnvtmbox** utility. To do this, run the following commands for each folder:

```
/usr/mmdf/bin/cnvtmbox old_folder new_folder
mv new_folder old_folder
```

For more information, see the **cnvtmbox** manual page.

Automatically forwarding mail

This section discusses **mail** features such as redirecting incoming messages to files other than */usr/spool/mail/username*, sending automatic responses, forwarding all your mail to another user, and using **mail** as a reminder service.

Redirecting incoming messages to other folders

SCO UNIX System V uses the MMDF Mail Transport Agent (MTA). You can use MMDF features to redirect specific messages to folders other than your system mailbox. Redirecting mail is useful for keeping all the mail on a particular subject or the messages sent to an alias to which you belong in one folder so that you can read them all at one time.

To tell MMDF to deliver mail to different folders, set up the special *.maildelivery* file in your home directory. Use the following procedure to set up *.maildelivery*:

1. Create the *.maildelivery* file in your home directory.
2. Make sure the permissions on the file are set to 644 (`-rw-r--r--`). To set the permissions, use the following command:

   chmod 644 ~/.maildelivery

3. Now, create a directory in which to put all the redirected mail. For examples, create a directory called *spool* in your home directory:

   mkdir ~/spool

   If your redirected mail directory is something other than *spool*, change all occurrences of spool in the examples to the new directory name.

   NOTE If you do not create the directory in which to redirect your mail before setting up *.maildelivery*, you can create an infinite loop as MMDF tries to deliver mail to a non-existent directory. Make sure you redirect mail to an existing directory!

4. Add the following lines to *.maildelivery*:

```
# Header        Content   Action   Result   Filename
to              login     >        ?        /usr/spool/mail/login
cc              login     >        ?        /usr/spool/mail/login
resent-to       login     >        ?        /usr/spool/mail/login
apparently-to   login     >        ?        /usr/spool/mail/login
```

   The number sign (#) is a comment character. Replace ***login*** with your login name.

These lines make sure all mail specifically addressed to you (mail with your name on the "To:", "Cc:", "Resent-To:", or "Apparently-To:" lines) is delivered to your system mailbox file.

The " > " character in the Action column tells MMDF to append the message to the specified file (*/usr/spool/mail/login*). The " ? " in the Result column tells MMDF to consider the message "delivered" if the action is successful. In other words, if the message arrives safely in the system mailbox, MMDF considers it "delivered." Note that " ? " prevents MMDF from carrying out the specified action if the message has already been delivered. This is important if you don't want MMDF to deliver a message to more than one folder.

5. Use this step to redirect mail addressed to an alias to which you belong. For example, if you want all the messages sent to the *engr* alias to go to a folder called *~/spool/engr.mail*, add the following lines to *.maildelivery*:

```
to          engr     >     A     /u/login/spool/engr.mail
cc          engr     >     A     /u/login/spool/engr.mail
```

The " A " in the Result column is identical to the " ? " in step 3, except that, even if the message is already considered "delivered," MMDF still performs the specified action.

MMDF creates the *engr.mail* folder when it first delivers mail to that location; you do not need to create the folder for MMDF to deliver the messages.

6. This step explains how to redirect messages with specific patterns in the "Subject:" header. To do this, simply specify "Subject" in the Header column and the pattern in the "content" column. For example, to redirect mail with the word "Style" in the "Subject:" header to *~/spool/style.mail*, add the following line to *.maildelivery*:

```
subject     "Style"  >     A     /u/login/spool/style.mail
```

7. Finally, add a line to catch and deliver any message that does not match the conditions you have specified. Add the following lines to *.maildelivery*:

```
default     -        >     A     /usr/spool/mail/login
#
```

The "default" in the Header column tells MMDF to match this field if the message has not already been delivered. The dash character (-) is simply a placeholder for an existing but unused field.

See "Saving a message" (page 234) for information on using the **folder** command to access the messages that you redirect to other folders.

Sending automatic responses when on vacation

MMDF includes a utility called **rcvtrip** that you can use to generate automatic replies to the messages that you receive while you are on vacation. MMDF still delivers your mail to your system mailbox (or other folders, if you have set up *~/.maildelivery* to redirect messages).

NOTE The **rcvtrip** utility sends an automatic response only to messages that specifically include your login name on the "To:" or "Cc:" lines.

In addition, **rcvtrip** keeps a record of who you have already sent a response to, and does not send another even if they send you more than one message while you are gone.

Use the following procedure to set up the appropriate files so that **rcvtrip** replies automatically to your incoming mail:

1. First, create a file called *tripnote* in your home directory. Make sure the permissions on the file are set to 644 (`-rw-r--r--`). To set these permissions, use the following command:

   ```
   chmod 644 ~/tripnote
   ```

2. Edit *~/tripnote* and add the response that you want MMDF to send to people when they send you mail. For example, you might want to tell people about where you are and when you will be back.
3. Create an empty file called *triplog* in your home directory using **touch**:

   ```
   touch ~/triplog
   ```

   MMDF uses this file to keep track of the people to whom you have already sent automatic responses.
4. Make sure that the permissions on *~/triplog* are set to 644:

   ```
   chmod 644 ~/triplog
   ```

5. Now, add the following as the first line in your *~/.maildelivery* file:[1]

```
*             -          pipe      R       rcvtrip $(sender)
```

NOTE You do not need to set up *.maildelivery* for redirecting messages if you just want to use **rcvtrip**. If this is the case, simply create the *.maildelivery* file in your home directory, add the **rcvtrip** line above, and make sure the permissions are set to 644.

This line tells MMDF to pipe all your messages to the **rcvtrip** program but does not consider the messages "delivered." In other words, MMDF still delivers the incoming messages to the appropriate folders. (See the **rcvtrip** manual pages for details.)

You should leave this line commented out (use the number (#) character) until immediately before you leave for your vacation.

6. When you return, comment out the **rcvtrip** line in your *.maildelivery* file.

The *triplog* file contains a list of all the people that sent you mail when you were gone. These lines list the address, date, and time you received the message, part of the "Subject:" line, and a plus (+) or minus (-) character to indicate whether or not **rcvtrip** sent an automatic response. You can use this file to check on who received automatic messages.

Forwarding your messages to another person

MMDF includes a utility called **resend** that allows you to forward all your incoming messages to another person automatically. For example, you might want someone else to read your mail when you go on vacation to ensure that any time-critical mail is answered promptly. As with **rcvtrip**, you use **resend** by adding a line to your *.maildelivery* file.[2]

1. For information on setting up your *.maildelivery* file to redirect messages to other folders, see "Redirecting incoming messages to other folders" (page 240).

2. For information on setting up your *.maildelivery* file to redirect messages to other folders or to call **rcvtrip**, see "Redirecting incoming messages to other folders" (page 240) and "Sending automatic responses when on vacation" (page 242).

Use the following steps to set up the *.maildelivery* file so that **resend** forwards your messages automatically:

1. If you have not already done so, create a file called *.maildelivery* in your home directory and make sure the permissions are set to 644.
2. Add the following as the first line in your *~/.maildelivery* file:

```
To              login    A     /usr/bin/resend     user_2
```

   Replace ***login*** with your address and ***user_2*** with the address of the person to whom you want to forward your messages.

This line in *.maildelivery* tells MMDF to redirect any mail with ***login*** in the "To:" header (mail addressed to you) to the **resend** utility which then forwards the messages to ***user_2***.

For more information on the format of the *.maildelivery* file, see the **maildelivery** manual page or "Redirecting incoming messages to other folders" (page 240). See the **resend** manual page for more information on the **resend** utility.

Using mail as a reminder service

Besides sending and receiving mail, you can use **mail** as a reminder service. Several UNIX system commands have this idea built in to them. For example, the **lp** command's **-m** option causes mail to be sent to the user after files have been printed on the lineprinter. When you log in, the operating system automatically examines the file named *calendar* in your home directory and looks for lines containing either today's or tomorrow's date. These lines are sent to you by **mail** as a reminder of important events.

If you program in the shell command language, you can use **mail** to signal the completion of a job. For example, you might place the following two lines in a shell procedure:

```
biglongjob
echo "biglongjob done" | mail login
```

You can also create a logfile that you want to mail to yourself. For example, you might have a shell procedure that looks like this:

```
dosomething > logfile
mail login < logfile
```

Summary of mail commands

The tables in this section summarize the **mail** commands and variables discussed in this chapter.

Table 24-1 mail Commands

Command	Abbreviation	Description
!		Executes a shell escape from command mode
alias	a	Creates personal mailing lists
delete	d	Deletes messages
dp		Deletes the current message and displays the next one
edit	e	Edits an existing message from command mode
exit	x	Exits **mail** without updating the folder
file	fi	Switches folders
folder	fold	Access messages saved in a file
forward	f	Forwards a message (indented) to another person
Forward	F	Forwards a message (not indented)
headers	h	Lists message headers at current prompt
	h+	Moves forward one window of headers
	h-	Moves back one window of headers
hold	ho	Holds messages in system mailbox
lpr	l	Sends messages to the lineprinter
preserve	pr	Preserves messages in system mailbox (see hold)
print	p	Displays non-deleted messages
quit	q	Updates the folder and quits **mail**
reply	r	Sends a reply to the author only
Reply	R	Sends a reply to everyone on the distribution list
save	s	Saves messages to folders
set		Sets **mail** options
shell	sh	Enters a new shell
undelete	u	Restores deleted messages
unset		Unsets **mail** options
visual	v	Edits an existing message from command mode
write	w	Writes messages to files

Table 24-2 Compose Mode Commands

Command	Description
~!	Executes a shell escape from compose mode
~?	Lists compose mode escapes
~b	Adds or changes the *Bcc:* header
~c	Adds or changes the *Cc:* header
~d	Reads in the *dead.letter* file
~h	Adds or changes all the headers
~m	Reads a message (indented) into the current message composition
~M	Reads a message (not indented) into the current message composition
~q	Aborts the current message composition
~r	Appends a file to your current message
~s	Adds or changes the *Subject:* header
~t	Adds names to the *To:* list

Table 24-3 mail Options

Option	Description
askcc	Prompts for carbon copy recipients
asksub	Prompts for a *Subject:* line
chron	Displays messages in chronological order, most recent last
folder	Defines the directory to hold your mail folders
hold	Keeps messages in the system mailbox when you quit
mchron	Displays messages in chronological order, most recent first
prompt	Sets the prompt to a different string
screen	Sets the size of a screen

Table 24-4 Other Commands

Command	Description
chmod	Changes the permissions on a file or directory
cnvtmbox	Converts old-style format mailboxes to MMDF format
mail	Sends a message or enter the **mail** utility
rcvtrip	Sends automatic responses to messages
touch	Creates and empty file
uuname	Determines which UUCP sites your computer communicates with

Table 24-5 Special Files

File	Description
calendar	Contains reminders for the operating system to mail to you
.maildelivery	Redirects incoming mail to other folders
triplog	Keeps track of the people you send automatic responses to
tripnote	Contains the message for **rcvtrip** to send automatically

Chapter 25
Sharing files between computers

File sharing lets you access files on a remote computer without copying files back and forth or logging in to the other computer. SCO Open Desktop/SCO Open Server provides two programs for this purpose: SCO NFS (this page) and LAN Manager Client (page 251). Ask your system administrator which program you should use.

Both NFS (Network File System) and LAN Manager make software on the network available through servers. A *server* is a computer that can provide its resources to other computers on the network. An individual workstation (called a *client*) can access information by requesting resources from a server through the network. A network can include many servers; and, under certain system configurations, a computer may be both a server and a client.

Sharing files under NFS

With NFS, you can access files on other computers as if they were on your computer. You do not need to enter any special commands or syntax.

The system administrator defines which remote filesystems are available to a client computer. Usually, available filesystems are mounted automatically. Occasionally, however, the system administrator designates a remote filesystem that you or another user must mount with the **mnt** command before you can access it. To find out whether such a filesystem is designated on your system, enter the following command:

```
mnt -t
```

You see entries like the following:

```
Mount Directory:               /cicely/w
Block Device:                  cicely:/w
Password required:             No
Mount if requested:            Yes
Fsck if requested:             No
   Fsck flags:                 -y
Mount at system startup:       No
Fsck at system startup:        No
   Fsck options:               -y
Run command before mounting:   No
Run command after mounting:    No
```

The line " Mount if requested: Yes " indicates that a user can mount this filesystem.

Mount the filesystem with this command:

mnt *filesystem*

Unmount the filesystem with this command:

umnt *filesystem*

If you want to see which filesystems are mounted on your computer, enter the **mnt** command without any arguments. You see a display similar to this:

```
Mounted /cicely/usr/man filesystem
Mounted /cicely/x filesystem
```

In this example, the filesystems */usr/man* and */x* from the server computer *cicely* are mounted on your computer. From your computer, you can access files from */usr/man* and */x* on the server *cicely* as if they resided on your computer. You have the same access permissions to these files that you would have if they were on your computer.

Although it might appear as though you now have your own copies of those files, the files are actually the very same files that users on the server access. When you add, modify, or remove files, NFS transparently performs your actions on the files on the server, and users working on the server see your actions immediately, as if you were working with the files on the server.

Sharing files under LAN Manager

With LAN Manager, you can access files, directories, programs, printers, and other resources on the following types of servers:

- LAN Manager for UNIX Systems server
- MS-NET server
- LAN Manager for OS/2 server
- XENIX-NET server

If you do not know which types of servers are on your network, or which resources are available, ask your network administrator.

Connecting to a server

NOTE Before you connect to a server, make sure you understand the information in "Naming files on non-UNIX systems" (page 253), "Understanding LAN Manager server security" (page 254), and "Getting network status information" (page 255).

To establish a connection to a remote file, printer, or other resource, use the **net use** command. To use this command, you must supply an alias, the server name, the resource name, and a password.

The alias is a short name that you choose, for referring to the resource after you connect. Do not choose an alias name that you are already using for another connection. Also, do not use the name of another node on the network as the alias. To find out the aliases of your current connections, enter the **net use** command (with no arguments) or the **net stat client** command. Ask your network administrator for a list of network node names to avoid.

The resource name and password refer to slightly different things for each type of server, and are explained in the following sections on connecting to each type of server. If you omit the password from the command line, you are prompted to enter it. This is a good method for keeping the password secure, because then it does not appear on the screen when you type it.

Connecting to a LAN Manager for UNIX Systems server

To connect to a LAN Manager for UNIX Systems server, enter this command:

net use *alias //server_name/resource_name* [*password*]

The *resource_name* is either a name that the system administrator has designated in an explicit share list, or the name of a user account on the server. The *password* is the password associated with the resource (if the server is in share mode), or the login password of either the client computer account or the LAN Manager guest account (if the server is in user mode).

Network

Ask your network administrator for the names of shared resources and the passwords you should use.

Connecting to an MS-NET server

To connect to an MS-NET server, enter this command:

net use ***alias //server_name/resource_name [password]***

The ***resource_name*** and ***password*** refer to entries in the MS-NET server's explicit share list. Ask your network administrator what resources are being shared and what their associated names and passwords are.

Connecting to a LAN Manager for OS/2 server

To connect to a LAN Manager for OS/2 server, enter this command:

net use ***alias //server_name/resource_name [password]***

The ***resource_name*** is a name that the system administrator has designated in an explicit share list. The ***password*** either comes from the explicit share list as well (if the server is in share mode), or is the password of the client computer account or the guest account (if the server is in user mode). Ask your network administrator what resources are being shared and what their associated names and passwords are.

Connecting to a XENIX-NET server from a non-trusted client

To connect to a XENIX-NET server from a non-trusted client, enter:

net use ***alias //server_name/account_name [password]***

The ***account_name*** and ***password*** are a user account on the server and the account's password.

Connecting to a XENIX-NET server from a trusted client

To connect to a XENIX-NET server from a trusted client, enter:

unix_command //server_name/pathname

You can use any supported UNIX system command. The ***pathname*** is the absolute pathname of the file or directory affected by the UNIX system command. Your permissions for the files and directories on the server are based on your user ID on the client.

To access a XENIX-NET server with the file-access permissions for another user (such as *root*), enter:

net use ***alias //server_name/account_name [password]***

The ***account_name*** and ***password*** are a user account on the server and the account's password.

Note that the alias you supply in the **net use** command refers to the *root* of the server's filesystem and not to the home directory of the account specified in that command.

Running commands on a server resource

Once you have connected to a server resource, run commands on it using the alias that you named in the net use command.

For example, you might use the following command to connect to a shared directory with the resource name of *database* on the server *cicely*:

```
net use db //cicely/database dbpass
```

(Here, ***dbpass*** is the password for the resource.)

You can then look at the file *report* in the shared directory by entering:

```
more //db/report
```

You can change to the shared directory by entering:

```
cd //db
```

Disconnecting from a server resource

Unless you are connected to a XENIX-NET server from a trusted client, disconnect from a server resource when you finish using it. Enter:

```
net use alias -d
```

Do not specify a resource name or password in a disconnect request.

Naming files on non-UNIX systems

In general, UNIX system filenames can contain both uppercase and lowercase characters and are case-sensitive. For example, the filenames *book.ch1*, *BOOK.CH1*, and *Book.CH1* refer to three distinct files on a UNIX system. When you use LAN Manager to access XENIX-NET servers, this case-sensitivity is preserved. However, when you access other types of servers with LAN Manager, the case-sensitivity is not preserved. Each server handles mixed-case filenames differently.

On MS-NET servers, filenames are mono-case, and appear all lowercase to users on a LAN Manager for UNIX Systems client.

On LAN Manager for OS/2 servers, filenames can be mixed-case. However, filenames appear in all lowercase letters to users on a LAN Manager for UNIX Systems client, even if the filenames are mixed-case on the LAN Manager for OS/2 server. To see the filenames as they appear on the server, log in to the server directly.

On LAN Manager for UNIX servers, files can be mixed-case, but only those filenames that are all lowercase are accessible to users on a LAN Manager for UNIX client.

Consequently, use only digits and lowercase letters in filenames when communicating with a LAN Manager for UNIX Systems, MS-NET, or OS/2 server.

The filenames you access through LAN Manager also have length limitations, depending on the type of server you access. With XENIX-NET servers, filenames are limited to the XENIX 14-character limit. With MS-NET, OS/2, or LAN Manager for UNIX Systems servers, filenames are limited to eight characters, followed optionally by a period and a one-, two-, or three-character extension. The extension should also consist only of lowercase letters or digits.

Understanding LAN Manager server security

Depending on the server, one of the following security schemes is in effect:

Security	Servers	Description
Share level	MS-NET LAN Manager for UNIX Systems LAN Manager for OS/2	Servers configured for share-level security are said to be in share mode. (MS-NET servers use only share-level security, so they do not have multiple security modes.) In share mode, you supply a resource name and password. For a resource on the explicit share list, supply the name and password assigned by the network administrator. For a user account (UNIX systems server only), supply the name of the account and the login password for that account.
User level	LAN Manager for UNIX Systems LAN Manager for OS/2	Servers configured for user-level security are said to be in user mode. In user mode, you supply the password for either your client machine account or the guest account. Your network administrator can tell you the correct password to use. For a resource on the explicit share list, supply the name assigned by the network administrator. For a user account (UNIX systems server only), supply the name of the account.

(Continued on next page)

(Continued)

Security	Servers	Description
(not applicable)	XENIX-NET	XENIX-NET servers do not use share-level or user-level security or explicit share lists. How you connect to a XENIX-NET server depends on whether your computer is a trusted or non-trusted client. To find out if your computer is a trusted client, ask your network administrator. For non-trusted access, you supply the login name and password for a user account on the server. For trusted access, you do not need to explicitly connect or supply login information.

Ask your network administrator for the names and passwords of shared resources, as well as the names of the servers where the resources are located.

Getting network status information

Use the **net use** and **net stat** commands to see information about current connections and processes.

Enter the **net use** command without additional arguments to display a list of current alias connections. The list looks similar to this:

```
Alias connections currently available are
  mo (owner maggieo) connected to server cicely
  rth (owner ruthanne) connected to server brick
  doc (owner joelf) connected to server rosalyn
```

Network

You can use the **net stat client** command to display a table of information about the network status of the local client processes. For example, when you enter **net stat client** on the computer *kbear*, you see a table like this:

```
LAN Manager Software installed on node kbear.

Consumer enabled.  Mon Jun 18 11:40:34 1990

Consuming from     VC State      dialect     vcid  nprocs  ntrans  bufsz
cicely             dormant       SCO Ext.       3       0      18     1k
  mo (owner maggieo) connected to server cicely
brick              active        SCO 1.3        6       2      71     3k
  rth (owner ruthanne) connected to server brick
rosalyn            active        core           4       1     651     1k
  doc (owner joelf) connected to server rosalyn
```

Consuming from	shows the alias, owner, and computer of origin of any remote connection. If no connection to a remote server exists, this part of the table is not shown.
VC State	shows if the connection between a client and server is actively in use at the time the **net stat** command is entered. • "aborted" means an error has occurred on this circuit. Usually, this circuit can be reestablished and used automatically. • "active" means that the circuit is currently in use; that is, a local process has a current directory or open file on the server. • "dormant" means no remote resources are in use at the moment. The VC state can change at any time. The **net stat** command provides only a snapshot of the network status at the moment the command is executed.
dialect	shows which SMB protocol dialect is being used over the network connection. For example, if this column lists the word "core," the Core Protocol is being used over the given connection.
vcid	is the ID number of the particular virtual circuit listed.
nprocs	is the number of client processes (for a "consuming" entry) that have open files or current directories on the remote server.
ntrans	is the number of request or reply transactions sent across the virtual circuit.
bufsz	is the size of the network buffer used for the connection.

Glossary

absolute pathname
A pathname that begins at the *root* (/) directory, for example, */etc/default/tar*. See also RELATIVE PATHNAME.

accelerators
Keystrokes that allow experienced users to select menu items using the keyboard. You can use accelerators to invoke a variety of menu items without using the mouse to select the item. Accelerators are listed to the right of their menu items.

accessory
A Desktop program that provides one of the accessory features of the Desktop, such as a text editor, mail, access to the UNIX system, and the help program.

account
The identification that you use to log in to a system. Accounts are set up by the system's administrator and identify the files and directories to which you have access.

active window
The window that currently accepts and displays mouse and keyboard input; it is identified by a special color. Only one window can be active at a time. If there is no active window, anything that you enter from the keyboard is ignored (except accelerator keystrokes).

address
The unique sequence of characters that identify a user in mail. On a single computer, the address is the user's account name. On a network of computers, the address includes the network address for the user's home computer in addition to the account name.

Administration control
This Desktop control lets system administrators configure and maintain system software from the Desktop. Display the Administration control by double-clicking on the Administration icon in the Controls window. The program's command line name is **sysadmsh**.

Administration icon
Start the Desktop Administration control by double-clicking on the Administration icon in the Controls window.

administrator
A person who manages a system or a network of systems. Typically, this person configures the system, maintains the network, assigns passwords and privileges, and helps users.

alias

In mail, a name that is equivalent to a group of user names. Aliases enable you to mail to more than one person using a single name.

application program

A program that performs some useful function, such as word processing, computer-aided design, or database management.

apply button

A button in some dialog boxes that implements your changes but does not save them for your next Desktop session. Click on the Apply button with mouse button 1 to apply your changes.

archiving

Storing files. See also BACKUP.

arrow keys

The four arrow keys on the keypad control cursor movement in text fields. In an application the effect of these keys may vary, but generally they move the cursor up, down, left, and right.

ASCII

The American Standard Code for Information Interchange is a standard way of representing characters on many computer systems. The term "ASCII file" is often used as a synonym for "plain text file," that is, a file without any special formatting, which can be viewed using UNIX system utilities such as **cat**, **more**, and **vi**.

awk

A simple programming language used principally for searching text files and extracting information from them. You can use **awk** as a powerful tool from the command line, or write short programs in it.

background process

A process that does not require interaction with the user to run. While a background process runs, the user can continue using other programs or commands. Background processing enables the operating system to execute multiple programs or commands at the same time.

backup

A copy of one or more files, directories, or filesystems that is stored apart from the original to safeguard against unplanned deletion. Used as a verb, it means to create a backup copy.

bell

The tone produced by a terminal or computer. The tone is sometimes used to indicate an invalid keystroke, an error, or a finished process.

bitmap

A picture or other graphical image that is stored as a series of bits (0's and 1's, each representing a single dot in the image). A bitmap can represent only two colors (usually black and white). Bitmaps are used to define the backgrounds displayed on the Desktop and in directory windows.

Bourne shell

The most widely used UNIX system shell. Most of the scripts supplied with SCO UNIX System V are written in the Bourne shell programming language. To start a Bourne shell from the command line, type **sh** and press ⟨Enter⟩.

buffer

An area of computer memory used to temporarily store information before it is written out to a more permanent location, like a file.

button

A button on a mouse or a graphical representation of a pushbutton on the screen. You can position the mouse pointer on an onscreen button and click to "push" the onscreen button and initiate the corresponding action.

C shell

An alternative shell supplied with SCO UNIX System V. This shell is known for its interactive features, such as the ability to recall and modify previous command lines. The C-shell shell programming language has a syntax like that of the C language. To start a C shell from the command line, type **csh** and press ⟨Enter⟩.

Calculator accessory

This Desktop accessory emulates a TI-30 or HP-10C scientific calculator and a slide rule. To display the calculator, double-click on the calculator icon in the Accessories window. The program's command line name is **xcalc**.

Calculator icon

To display the Calculator accessory, double-click on the calculator icon in the Accessories window.

cancel button

A button in some dialog boxes that closes the dialog box without implementing any changes. Click on the Cancel button with mouse button 1 to cancel any changes you may have made.

carriage return

The signal sent when the key labeled "Return" or "Enter" is pressed. It is represented as ⟨Enter⟩ in this documentation. It usually indicates that you have finished typing in a text field, that you have reached the end of a line of text, or that you are accepting text entered in a dialog box.

cascading menu

A submenu that displays when you select certain menu items. A cascading menu provides additional menu items.

checkbox

An onscreen button that lets you select or deselect an option by clicking mouse button 1. When the checkbox is highlighted or colored, that option is selected. Checkboxes are used when you can select none, one, or more than one of a group of options. See also RADIO BUTTON.

class

A file's class is indicated by a group of one to six characters that show its type, permissions, and ownership. For more information on classes, see the *System Administrator's Guide.*

click

To press briefly and release a mouse button. You click a mouse button; you click *on* an icon (by pointing to it and clicking the mouse button). When the mouse button is not specified, click refers to mouse button 1. See also DOUBLE-CLICK.

clipboard

A place in computer memory where text is temporarily stored during a cut-and-paste operation.

Clock accessory

This Desktop accessory displays an analog or digital clock showing the current time. To display the clock, double-click on the Clock icon in the Accessories window. The program's command line name is **xclock**.

Clock icon

To display the Clock accessory, double-click on the Clock icon in the Accessories window.

Color control

This Desktop control lets you change the color scheme for all Desktop windows. Start Color by double-clicking on the Color icon in the Controls window. You can choose from a list of supplied color schemes or create your own. The program's command line name is **scocolor**.

Color icon

Start the Desktop Color control by double-clicking on the Color icon in the Controls window.

command

A command is a series of words or characters that send an instruction or request to the computer.

command line

A place on your screen where you can type commands. You use command lines to communicate with the operating system in a UNIX or DOS window. See also PROMPT.

command separator

The semicolon (;). To issue several commands on one UNIX command line, separate the commands with semicolons.

computername
The name by which the computer is known on the network. Each computername on the network must be unique.

context-sensitive help
A facility that provides information specific to the command, window, or part of the screen that you are currently using.

control key
The control key is marked "Ctrl" on most terminals and is used as a modifier key. It is represented as ⟨Ctrl⟩ in this documentation. The control key can be compared with the ⟨Shift⟩ key on a typewriter keyboard, because it is always used in conjunction with another key or mouse button. This is done by pressing and holding down the control key, pressing the required character or mouse button, then releasing both.

control character
A character in a file that was typed using the ⟨Ctrl⟩ key in conjunction with another key.

control
A Desktop program that modifies or controls some aspect of your Desktop environment, such as colors, fonts, or startup behavior.

current directory
The directory where you are currently working. The current directory is taken as the starting point for all relative pathnames.

cursor
A blinking or constant box, underline, I-shaped character or other graphical image that shows your position in text. See also MOUSE POINTER.

cut and paste
To move a block of text from one position to another, either within a file or between files or windows. The text is temporarily stored on a clipboard during the cut-and-paste operation.

cut
To remove a marked area of text to a clipboard so it can be inserted elsewhere. See also CUT AND PASTE.

default
The standard configuration or values for a screen, program, or field. For example, the default Desktop is displayed the very first time you start the Desktop, before you have made any changes.

Desktop
A graphical user interface for communicating with the computer.

Desktop menu bar
A menu bar displayed at the top of the Desktop. Click on one of the options, **File, Edit, View,** or **Help,** to display the corresponding menu. These Desktop menus provide options for manipulating the icons on the Desktop or the appearance of the Desktop.

device
Peripheral hardware attached to the computer such as printers, modems, disk and tape drives, terminals, and so on.

dialog box
A pop-up window that contains options or instructions. By responding, you carry on a kind of "dialog" with a program.

dimmed
Displayed with reduced intensity or stippled characters. This convention is used to indicate that a button, control, or menu item is currently unavailable.

directory
A special type of file that contains a collection of files and other directories. Directories are part of a hierarchical system of organizing files and can be used to group related files and directories together.

directory icons
The icons that represent directories. To open a directory window, double-click on the directory icon with mouse button 1. The directory window displays the files and any subdirectories contained in the directory.

directory menu bar
A menu bar displayed at the top of every directory window, directly beneath the title bar. Click on one of the options, **File, Edit, View,** or **Help,** to display the corresponding menu. These directory menus provide options for manipulating the icons in that directory or the appearance of the directory window. The directory menus in the Accessories, Controls, Applications, and Trash windows provide an abbreviated list of options similar to those provided on other directory windows.

directory window
The window that displays the files and any subdirectories contained in a directory. Open a directory and display its directory window by double-clicking on the directory icon with mouse button 1.

diskette
A thin, flexible data storage disk permanently enclosed in a protective jacket. It is also called a floppy disk.

DOS
The operating system most commonly used on small personal computers.

DOS command line
A line in a DOS window, on which you can enter commands to communicate with the DOS operating system.

DOS drive

The personal computer hardware associated with DOS diskettes or fixed disks. DOS drives are named with letters followed by a colon. By convention, diskette drives are usually designated A: and B:, and a local fixed disk is designated C:.

DOS executable files

A file that contains a DOS program or command. The program or command is executed when you enter the name of the file on a command line. DOS requires names of executable files to end with one of the three-letter extensions .BAT, .COM, or .EXE (for example AUTOEXEC.BAT and COMMAND.COM.)

DOS icon

Open a DOS window by double-clicking on the DOS icon in the Accessories window.

DOS window

A window that allows you to communicate directly with the SCO Open Desktop/SCO Open Server emulation of the DOS operating system.

dot file

A UNIX file that configures some aspect of the Desktop, related programs, or your computer environment. Dot files are so called because their file names start with a period or "dot." They usually are created automatically and reside in each user's home directory. You can choose whether or not you want to display dot files in a directory window.

double-click

To click a mouse button twice in rapid succession. When the mouse button is not specified, double-click refers to mouse button 1. See also CLICK.

drag

To press and hold a mouse button while moving the mouse, which moves a selected object on the screen. If you select multiple icons, dragging one of them performs the same action on all of them. See also DROP.

drop

To drag an icon on top of another and release the mouse button. You drop one icon on another to perform certain tasks, such as starting the text editor or sending a file to a printer.

editor

A program you use to edit (write or change) text stored in a file.

Edit accessory

A Desktop accessory used to create and edit text files. Start the editor by double-clicking on the Edit icon or by dropping a file icon on the Edit icon. The program's command line name is scoedit.

Edit icon

Start the Desktop editor by double-clicking on the Edit icon or by dropping a text file icon on the Edit icon.

electronic mail

Messages sent via the computer. You can send messages to any other user on your system. You may be able to send messages to users on other computers if your computer is part of a network. See also MAIL ACCESSORY and MAIL ICON.

enter

To input one or more characters from the keyboard, followed by a carriage return. For example, "Enter your password" means to type the characters of your password and then press the ⟨Enter⟩ key.

enter key

See CARRIAGE RETURN.

environment

The various settings that control the way you work on the UNIX system. These settings are specific to the shell you use and can be modified from the command line or by modifying shell control files. For example, the directories the shell searches to find a command you type are set in the variable **$PATH**, which is part of your environment.

escape key

The escape key is marked "Esc" on most keyboards. In this documentation, the escape key is referred to as ⟨Esc⟩.

executable file

A file that contains a program or command. To run the program or command, double-click on the associated icon, drop an appropriate file on the icon, or enter the filename at the prompt in a UNIX window. Some executable files are in binary form and are not readable like regular text files. See also DOS EXECUTABLE FILES.

executable file icons

The icons that represent executable files. Double-clicking on an executable file icon executes the program or command represented by that icon. Some programs have their own specially designed icons and others use a generic executable file icon.

file

A named collection of information stored on your computer. A file can contain regular text, commands, a program in binary form, or it can contain special characters in a format that is recognized by a specific program. See also DIRECTORY.

filename

The name of a file, which must be unique in the directory in which it resides.

file properties

The information that can be displayed about an icon, such as its name, owner, group, size, type, class, the number of links it has, its access times, and the permissions assigned to it.

filesystem
A hierarchical organization of directories and files. In a filesystem, each file has a unique location in relation to all others. See also PATHNAME.

Find accessory
A Desktop accessory that searches within your home directory and its subdirectories to find a specified file. You can find a file by selecting **Find** from the **Desktop File** menu or by double-clicking on the Find icon in the Accessories window.

Find icon
Start the Desktop Find accessory by double-clicking on the Find icon in the Accessories window.

floppy disk
See DISKETTE.

font
A style of text characters on the Desktop or elsewhere.

font angle
The angle of the font, such as *italic* or regular.

font family
A group of closely related fonts that differ in weight, angle, or size.

font path
The pathname that identifies where the font description resides.

font point size
The nominal size of font characters in points. There are 72 points to an inch.

font weight
The heaviness of the font, such as **bold** or regular.

ftp
A file transfer program that allows you to copy files to and from a remote computer in a network.

function keys
The keys marked ⟨F1⟩ through ⟨F10⟩ (sometimes more) on your keyboard. These keys are used in some programs to issue specific commands.

general font
The font used in menus and messages on the Desktop.

ghost icon
An icon that represents a file that cannot be located. Putting a ghost icon away removes it from the Desktop for the current session.

grayed selection
See DIMMED.

group

A set of users on a computer. Every user belongs to one or more groups. User groups are usually comprised of users with similar job functions and similar needs for access to files. Each file on the UNIX system also has a group associated with it; this group, along with the owner and the permissions of a file, controls who can access and modify that file.

Help accessory

A Desktop accessory (also accessible from most controls and accessories) that provides explanations, assistance, or more information. It is also called *online help.*

Help icon

Start the Help program by double-clicking on the Help icon. To get information about a program, drop its icon icon on the Help icon.

help key

The function key that calls online help (⟨F1⟩ by default).

Help menu

The list of options displayed when you invoke online help. Depending on the context, the menu includes one or more of these options: **On Help, On Window, On Keys, On Context, On Object, On Version, On Index,** and **Tutorial.**

home directory

The place in the filesystem where you can keep your personal files and subdirectories. When you log in, you are automatically placed in your home directory.

host

Any computer on a network.

icon

A graphical representation of a file, directory, program, or window.

icon font

The font used in icon labels.

icon label

The name displayed with the icon.

icon picture

The graphical image part of an icon.

iconify

To store a window as a window icon (also called "minimize"). When you iconify a window, the only indication that it is open is its icon.

inactive window

A window that does not currently accept mouse or keyboard input. See also ACTIVE WINDOW.

kernel
The central part of the UNIX operating system, which manages how memory is used, how tasks are scheduled, how devices are accessed, and how file information is stored and updated.

Korn shell
The Korn shell is compatible with the Bourne shell, but provides a much wider range of programming features. The Korn shell also offers improved versions of many of the C shell's interactive features. To start a Korn shell from the command line, type **ksh** and press ⟨Enter⟩.

LAN
Local Area Network. On a LAN, high-speed cable (usually an Ethernet cable) directly connects the computers at a single site. See also WAN.

link
A filename that points to another file. Links let you access a single file from multiple directories without storing multiple copies of the file. If you make a change to the content of a linked file, the change is reflected in each of the links.

Load accessory
A Desktop accessory that displays a histogram (graph) of the system load average. The histogram is updated periodically. The program's command line name is **xload**.

Load icon
Start the Desktop Load accessory by double-clicking on the Load icon in the Accessories window.

local
On a network, the computer that you originally logged in to. See also REMOTE COMPUTER.

Lock control
A Desktop control that locks the display so no one can use your terminal until you enter your password. The program's command line name is **scolock**.

Lock icon
Lock your terminal by double-clicking on the Lock icon in the Controls window.

locked icon
An icon that cannot be removed from the Desktop window except by the system administrator.

log in
To enter your user name and password so you can gain access to the computer.

login
The process of logging in or the name you use when logging in.

log out
: To end a computer session and return to the login screen or prompt.

lower
: To move a window underneath all the other open windows on the Desktop.

lp
: A program that prints files.

macro
: A combination of keystrokes in the **vi** editor that represent, and provide a quick way of entering, a series of editing commands. Create macros and abbreviations in **vi** to automate work you do regularly.

Mail accessory
: A Desktop accessory that sends and delivers electronic mail. The Mail accessory provides tools for composing your messages and for manipulating messages you receive. The program's command line name is **scomail**.

mail alias
: A single name used to send mail to several users at once. For example, many companies have aliases set up for mailing to the entire company, single departments, or groups of individuals.

mail folder
: A directory in which you can store related mail messages.

Mail icon
: Start the Desktop Mail accessory by double-clicking on the Mail icon.

manual pages
: The reference documentation for individual commands, utilities, and programs. They are also called "man pages."

maximize
: To enlarge a window so that it fills the entire screen.

maximize button
: A button in the upper right corner of a window frame that maximizes the window. The button contains an image of a square.

menu
: An onscreen list of items from which users can select. Selecting an item causes an action to be taken or another menu to be displayed. Menus are usually located at the top of the window frame in a menu bar.

menu bar
: A bar across the top of a window, under the title bar, that contains the titles of the pull-down menus for that window.

menu item
: A choice on a menu.

message box
A dialog box that provides information, gives the current state of work in progress, asks a question, issues a warning, or draws attention to an error.

message list window
A window in the Mail program that displays the list of mail messages received.

metacharacter
See WILDCARD.

minimize
To store a window as a window icon, temporarily removing the window display. A program running in a window continues to run in the background when you minimize the window. This is also called "iconifying."

minimize button
A button in the upper right corner of a window frame that minimizes the window. The button contains an image of a small square.

missing-picture icon
An icon that is used when the correct icon picture cannot be located. The icon functions normally even though the correct picture is not displayed.

Motif window manager
A program that controls window configuration and creates window frames. It is also called **mwm**.

mouse
A pointing device used to move a pointer about on the screen. See also MOUSE BUTTON.

mouse button
One of the buttons on a mouse pointing device. The mouse buttons select or manipulate graphical objects. See also CLICK, DOUBLE-CLICK, DRAG, and DROP.

mouse button 1
The mouse button that you click to make a selection or to move or copy icons. Button 1 is usually the leftmost button, but this can be changed using the Mouse control.

Mouse control
A Desktop control used to change mouse characteristics. Start the Mouse control program by double-clicking on the Mouse icon in the Controls window. The program's command line name is **scomouse**.

Mouse icon
Change mouse characteristics by double-clicking on the Mouse icon in the Controls window.

mouse pointer

The graphical image, such as an arrow or other symbol, that indicates your current position on the screen. Move the pointer by moving the mouse. The pointer changes shape to indicate what you can do at a particular location or to indicate a program's status. For example, if a program is busy, the pointer may change to an image of an hourglass.

network

A group of computers that are linked together and can communicate with each other.

normalize

To restore an iconified window to its "normal" (original) appearance.

OK button

A pushbutton in dialog boxes that accepts any changes you made in the box. Click on the OK button with mouse button 1 to accept your changes.

online help

A facility in the Desktop and in some control, accessory, and application programs that provides explanatory text if you need assistance or more information.

open

To display a file or window so you can view or modify its contents. Use the Desktop to open windows; use a program to open files.

operating system

A group of programs that provide basic functionality on a computer. These programs operate your computer hardware in response to commands like **copy** and **print**, and form a set of functional building blocks upon which other programs depend. An operating system also manages computer resources and resolves resource conflicts, as when two programs want to use a disk drive at the same time.

owner

The user who created a file or directory. Only the owner and the super user can change the permissions assigned to the file or directory.

Paint accessory

A Desktop accessory used to create and edit graphical images. Start Paint by double-clicking on the Paint icon in the Accessories window. The program's command line name is **scopaint**.

Paint icon

Start the Desktop Paint accessory by double-clicking on the Paint icon in the Accessories window.

palette

A collection of colors or palettes. Palettes are used by the Desktop Color control and the Desktop Paint accessory.

parent directory icon
An icon that moves you up one directory in the filesystem hierarchy when you double-click on it.

password
A confidential sequence of characters that you use to confirm your identity to the computer when you log in or unlock a locked display.

paste
To insert previously cut or copied text from the clipboard into a file or a window.

pathname
A list of the directories and subdirectories that define the precise path to a file from the *root* directory (/). See also RELATIVE PATHNAME.

pattern match
To find sequences of characters in a text file that match those you specify.

permissions
The settings (also called properties or attributes) that regulate which users can access each file and directory, and what types of access are permitted.

pipe
A way of joining commands on the command line so that the output of one command provides the input for the next. To use a pipe on the command line, join commands with the pipe symbol, " | ".

pixmap
A picture or other graphical image that can include multiple colors.

point
To position the pointer.

pointer
See MOUSE POINTER.

pop-up window
A window that is displayed on top of the active window. Pop-up windows display choices, ask for user input, or display informative text.

Preferences control
This Desktop control changes Desktop appearance, icon and cursor types, fonts used on the Desktop, and Desktop background patterns. Display the Preferences control by double-clicking on the Preferences icon in the Controls window.

Preferences icon
Display the Desktop Preferences control by double-clicking on the Preferences icon in the Controls window.

Print accessory

A Desktop accessory that prints text files. To print a text file, drop its icon on the Print icon.

Print icon

Print a file by dropping its icon on the Print icon.

print queue

A queue (line) in which print requests are stored while they are waiting to be printed.

print spooler

A program that manages print requests.

process ID

A number that uniquely identifies a running program on the UNIX system. This is also known as the PID.

program

Software that performs a task. Desktop controls, accessories, and applications are all programs.

prompt

One or more characters or symbols that identify a line on which commands can be entered, as in a UNIX or DOS window. "Prompt" also refers to the text displayed when the computer displays a request for input or an instruction.

pull-down menu

A menu that can be pulled down from the menu bar by clicking on its name.

pushbutton

A graphical representation on the screen of a real-life pushbutton. You can position the mouse pointer on an onscreen button and click to "push" the onscreen button, thus initiating the corresponding action.

radio button

An onscreen button used for switching an option off or on. When the radio button is highlighted or colored, that option is selected. Radio buttons are used when one (and only one) of a group of options must be selected. See also CHECK BOX.

rcp

A file transfer program for copying files to and from a remote computer in a network.

regular expression

A notation for matching any sequence of characters. Many UNIX system utilities use regular expressions including the vi editor and the awk programming language.

relative pathname
A pathname that does not start with a slash (/); for example; *Tutorial*, *Reports/September*, or *../tmp*. A relative pathname is searched for starting from the current working directory and may use the notation " .. " to indicate "one directory up from the current working directory." See also ABSOLUTE PATHNAME.

remote computer
A computer in a network other than the computer that you originally logged in to.

request-ID
A number assigned to each print job when you send it to the printer. The number is displayed when you print a file.

resize
To change the height and/or width of a window.

restore
To return an iconified or maximized window to its normal size. Also, to copy files from a backup onto the system.

rlogin
A program for logging in to a remote computer.

root
The top directory of a UNIX filesystem, represented as a slash (/). Also, the login name of the super user, a user who has the widest form of computer privileges.

root directory
The top directory of a UNIX filesystem, represented as a slash (/). Root is the home directory of the super user.

Root menu
A standard menu that displays when you press and hold mouse button 1 on the edges of the Desktop or in the Root window. The Root menu lets you shuffle windows, refresh your screen, restart the window manager, and log out.

Root window
The background on your terminal screen. The Desktop that appears when you first log in is in the Root window. If you use the "Preferences" control option to turn off "Desktop as backdrop," the background seen behind the Desktop is the Root window.

scocolor
See COLOR CONTROL.

scoedit
See EDIT ACCESSORY.

scohelp
See HELP ACCESSORY.

scolock
See LOCK CONTROL.

scologin
The program that displays the SCO Open Desktop/SCO Open Server login box.

scomail
See MAIL ACCESSORY.

scomouse
See MOUSE CONTROL.

scopaint
See PAINT ACCESSORY.

scosession
See SESSION CONTROL.

scoterm
See UNIX WINDOW.

screen
The computer's display that shows your communications, or "input," to the computer. It also shows some of the computer's "output," such as prompts and the results of your commands.

scroll
To change your viewpoint in a window or file, either vertically or horizontally, thus displaying information that is not currently visible.

scroll bars
The sliding controls located on the right or bottom of some windows, which you move to scroll through the window contents.

select button
See MOUSE BUTTON 1.

select
To choose a menu item, list item, or an icon by clicking on it with mouse button 1. To select multiple icons, hold down the ⟨Ctrl⟩ key while clicking on each of them with mouse button 1.

Session control
A Desktop control used to change the startup and shutdown behavior of the software. Start the Session control program by double-clicking on the Session icon in the Controls window. The program's command line name is **scosession.**

Session icon
Start the Session control program by double-clicking on the Session icon in the Controls window.

shell
A program that controls how you interact with the operating system. Using such programs, you can write your own "shell scripts" to automate work you do regularly. The shells available with SCO UNIX System V include the Korn shell, the Bourne shell, and the C shell.

shell escape
A command you type from within an interactive program to escape to the shell.

shell programming language
A programming language that is built into the shell. The Korn shell, the Bourne shell, and the C shell all have slightly different programming languages but all three shells offer basics such as variable creation, loops, and conditional tests.

shell script
An executable text file written in a shell programming language. Shell scripts are made up of shell programming commands mixed with regular UNIX system commands. The shell running the script will read it one line at a time and perform the requested commands.

shuffle
To move a window up or down through a stack of open windows on your screen.

stack
A group of open windows, one on top of another.

standard input
The usual place from which a program takes its input. By default, this is the keyboard. Standard input can be redirected; for example, you can use the less-than symbol (<) to instruct a program to take input from a file.

standard output
The usual place to which a program writes its output. By default, this is the screen. Standard output can be redirected; for example, you can use a pipe symbol (|) to instruct a program to write its output into a pipe, which will then be read as input by the next program in the pipeline. Or, you can use " > " to redirect the standard output to a file.

subdirectory
A directory that is located inside (below) another directory.

submenu
A menu that appears when a menu item is selected. It provides additional choices specific to the selected item. It is also known as a cascading menu.

superuser

A user who has powerful special privileges needed to help administer and maintain the system. The superuser logs in as root.

system administrator

The person who looks after the day-to-day running of the computer and performs tasks such as giving new users accounts and making system backups.

telnet

A program that communicates with another computer in a network.

terminal

Video display unit with a keyboard, a monitor, and sometimes a mouse. These units differ from computers in that they do not have any real processing power themselves; they must be connected to a computer before they can do any useful work.

text file

A file that contains text.

title bar

The bar at the top of the window frame that contains the window's title or name. In certain programs' windows, the title bar can contain the name of a file that the program has opened. You can move the entire window by dragging the title bar.

toggle

To switch back and forth between any two conditions. For example, to toggle from YES to NO on a menu.

trash directory

A subdirectory of a user's home directory in which files are temporarily stored before being removed from the system permanently.

Trash icon

A Desktop icon that represents the trash directory. You can remove files by dropping them on the Trash icon.

type

The category that describes whether the file is a regular file, a directory, or other type of file, such as a block device or character device. For more information on file types, see the *System Administrator's Guide.*

UNIX

A type of operating system. SCO Open Desktop/SCO Open Server is based on the UNIX operating system.

UNIX command line

A line in a UNIX window on which you can enter commands to communicate with the UNIX operating system.

UNIX icon
Open a UNIX window by double-clicking on the UNIX icon.

UNIX window
A Desktop accessory that allows you to communicate directly with the UNIX operating system.

user name
The name by which a user is known in a UNIX operating system.

variable
An "object" that stores a particular value. The value of a variable can be changed either from inside a program or from the command line. Each shell variable controls a particular aspect of your working environment on the UNIX system. For example, the variable **PS1** stores your primary prompt. By default, this variable has a value of " $ " for Bourne shell users.

WAN
Wide Area Network. On a WAN, telephone lines connect computers at one site to computers at another site. See also LAN.

wildcard
A character (such as " ? " or " * " that stands for another character or a group of characters in text searches and similar operations. Also called "metacharacters."

window
A rectangular area on the screen that is dedicated to specific tasks. Windows display information such as the files and subdirectories in a directory; all or part of the text in a file; prompts for information that is required by a program; status information; or data that is being modified.

window frame
The graphical borders around a window, including the title bar, menu bar, scroll bars, the minimize and maximize buttons, and the window menu button.

window icons
The icons that represent minimized windows.

window manager
A program that controls window configuration and creates window frames. SCO Open Desktop/SCO Open Server uses the Motif window manager (mwm).

window menu

A standard menu that appears when you click on the window menu button. Use the Window menu to restore, move, resize, minimize, maximize, lower, and close the window.

window menu button

The button at the top left corner of a window. When pressed, it displays the **Window** menu for that window. You can double-click on the window menu button to close the window.

Index

SCO®
OPEN SYSTEMS SOFTWARE

Please help us to write computer manuals that meet your needs by completing this form. Please post the completed form to the Publications Manager nearest you: The Santa Cruz Operation, Ltd., Croxley Centre, Hatters Lane, Watford WD1 8YN, United Kingdom; The Santa Cruz Operation, Inc., 400 Encinal Street, P.O. Box 1900, Santa Cruz, California 95061, USA or SCO Canada, Inc., 130 Bloor Street West, 10th Floor, Toronto, Ontario, Canada M5S 1N5.

Volume title: ______________________________

(Copy this from the title page of the manual)

Product: ______________________________

(for example, SCO UNIX System V Release 3.2 Operating System Version 4.0)

How long have you used this product?

❑ Less than one month ❑ Less than six months ❑ Less than one year

❑ 1 to 2 years ❑ More than 2 years

How much have you read of this manual?

❑ Entire manual ❑ Specific chapters ❑ Used only for reference

	Agree				*Disagree*
The software was fully and accurately described	❑	❑	❑	❑	❑
The manual was well organized	❑	❑	❑	❑	❑
The writing was at an appropriate technical level (neither too complicated nor too simple)	❑	❑	❑	❑	❑
It was easy to find the information I was looking for	❑	❑	❑	❑	❑
Examples were clear and easy to follow	❑	❑	❑	❑	❑
Illustrations added to my understanding of the software	❑	❑	❑	❑	❑
I liked the page design of the manual	❑	❑	❑	❑	❑

If you have specific comments or if you have found specific inaccuracies, please report these on the back of this form or on a separate sheet of paper. In the case of inaccuracies, please list the relevant page number.

May we contact you further about how to improve SCO documentation? If so, please supply the following details:

Name ____________________ *Position* ____________________

Company ______________________________

Address ______________________________

City & Post/Zip Code ______________________________

Country ______________________________

Telephone ____________________ *Facsimile* ____________________