Commodore®

Amiga® UNIX®
System V Release 4

Using Amiga UNIX

Commodore and Amiga are registered trademarks of Commodore Electronics Ltd. and Commodore-Amiga, Inc., respectively. This document may also contain reference to other trademarks and registered trademarks for the various products listed which are believed to belong to the sources associated therewith.

The information contained herein is subject to change. Furthermore, this listing should not be considered as containing any endorsement or representation with respect to the products listed, their use, results, performance, capabilities, appropriateness or availability.

THIS INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED. THE ENTIRE RISK AS TO THE USE OF THIS INFORMATION IS ASSUMED BY THE USER.

IN NO EVENT WILL COMMODORE BE LIABLE FOR ANY DAMAGES, DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL, RESULTING FROM ANY DEFECT IN THE INFORMATION, EVEN IF IT HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

If this product is being acquired for or on behalf of the United States of America, its agencies and/or instrumentalities, it is provided with RESTRICTED RIGHTS, and all use, duplication, or disclosure with respect to the included software and documentation is subject to the restrictions set forth in The Rights in Technical Data and Computer Software clause at 252.227-7013 of the DOD FAR and the Rights in Data-General clause at 52.227-19 of the FAR. Unless otherwise indicated, the manufacturer/integrator is Commodore Business Machines, Inc., 1200 Wilson Drive, West Chester, PA 19380.

OPEN LOOK is a trademark of AT&T.
UNIX is a registered trademark of AT&T.
Ethernet is a trademark of Xerox Corporation.
PostScript is a registered trademark of Adobe Systems, Inc.
X Window System is a trademark of the Massachusetts Institute of Technology.
XENIX and MS-DOS are registered trademarks of Microsoft Corporation.

Preface

This manual is written for anyone who wants to use Amiga UNIX. It includes conceptual introductions to topics, instructions for common user and administrative tasks, and a reference section covering most major commands.

Beginning users may want to start with a simpler book, such as *Learning Amiga UNIX*. Advanced users and administrators of large systems may want to supplement this book with selections from UNIX Press.

Credits

Written by Kendall Robinson and Carol Wahl

Directed by Richard Buck

Reviewed by Jan Carlson, Mike Ditto, Ken Farinsky, Mike Hall, and Rich Skrenta

Artwork by Cynthia Robinson

Additional contributions by Dave Ballman, Don Bein, Jesse Bornfreund, Paul Calkin, Keith Gabryelski, Sam Frederick, and Randy Gort

Produced by Johann George

This manual was written and produced using Amiga UNIX and the X Window System. 11/90

Table of Contents

Using this manual

Using this manual

Who should read this manual?

This manual is directed at two audiences:

- newcomers to UNIX who want an introduction to some of the most important features and commands
- experienced UNIX users who want to learn about new Amiga UNIX features

What kind of book is it?

Most of this book is written as an introductory reference guide, providing a little information about a large variety of common tasks. No previous knowledge is required; you can read any page at any time, and it should tell you what you need to get started with a task or command. If you need more advanced information, there are many other sources you can check. This manual is not intended to provide advanced information.

Basic chapters

The first six chapters cover basic tasks that any new UNIX user will encounter at some point (and which an experienced UNIX user will already know). The subjects include files and directories, online help, simple networking, printing, editing, and electronic mail. These chapters apply to many UNIX systems, not just Amiga UNIX, and could be used as an introductory guide for any UNIX System V Release 4 system.

Advanced chapters

The chapters become increasingly complex as you advance through the book; the second half (chapters 7 through 11) present information that many novice users will not need to know. The final chapter is specifically a reference chapter. It covers many commands in a bit more detail than the rest of the book.

Who should read which chapter?

Chapter	What does it cover?
1, 2	Basic information for new users: getting started, online help
3, 4, 5, 6	Basic information for all users: files and directories, printing, editing in vi, using elm to send or read mail
7	Overview of multi-user topics: multiple users, multiple login sessions, networking, exchanging files and messages, checking status
8, 9, 10	Advanced information: details of new Amiga Unix features and commands, system files, system maintenance overview for people who manage their own systems
11	Reference chapter for all users

Terms

Most of the terms in this document are explained where they are used; since each UNIX command tends to be its own odd form of abbreviated English, a glossary of terms would be a list of commands, or even a complete index. In general, no terms are used in a way that differs from standard UNIX or computer usage.

Press the RETURN key

We do not list the RETURN key each time it should be pressed; we assume that users either know or will quickly learn to end a command by pressing RETURN.

Conventions

Specific typographic conventions are used to either identify characteristics or focus your attention. These conventions are minimized so the manual reads as smoothly as possible. We use four such conventions, to show command lines in text, substitutions, keys, and the appearance of a screen.

Four typographic conventions

We use four typographic conventions in this manual to highlight specific concepts.

Convention	What does it mean?
bold words	a Unix command or command line: **ls -lt** **man ls**
italics	a value to be substituted, frequently as part of a command line: *filename* **cat *filename*** also used for chapter and manual titles *Using Amiga Unix* *Learning the Basics*
uppercase	a key on the keyboard, or a combination of keys if connected by a hyphen: RETURN CTRL-C
typewriter font	text that appears on your screen either as you type it or in response to your commands; values to be substituted by you are in italics

Notes for users

Notes for experienced UNIX users

Experienced UNIX users already know most or all of the information covered in the first six chapters; it's straight UNIX, and the only parts you might not know are the System V Release 4 enhancements from AT&T and the unique additions of Amiga UNIX. Concentrate your attention on chapter 10, Maintaining your system (in case there are system files and administrative procedures that are new to you) and chapter 8, *Special features of Amiga UNIX* (which documents all the new Amiga UNIX commands).

Notes for PC users

Your PC experience will make you a quick learner, but it may not help you to guess which UNIX command serves which purpose. (Once you become accustomed to the names and conventions, UNIX will seem easy; until then, you probably won't find many familiar-sounding commands.)

As an experienced PC user, you might be able to skip the introductory chapters written for new users. However, these chapters are short, and they specifically focus on common computer tasks and how they are performed in UNIX. Since the tutorials in Learning to Use Amiga UNIX were written specifically to help PC users transfer to UNIX, we recommend that you read that book and then scan the first few chapters of this manual.

PC commands

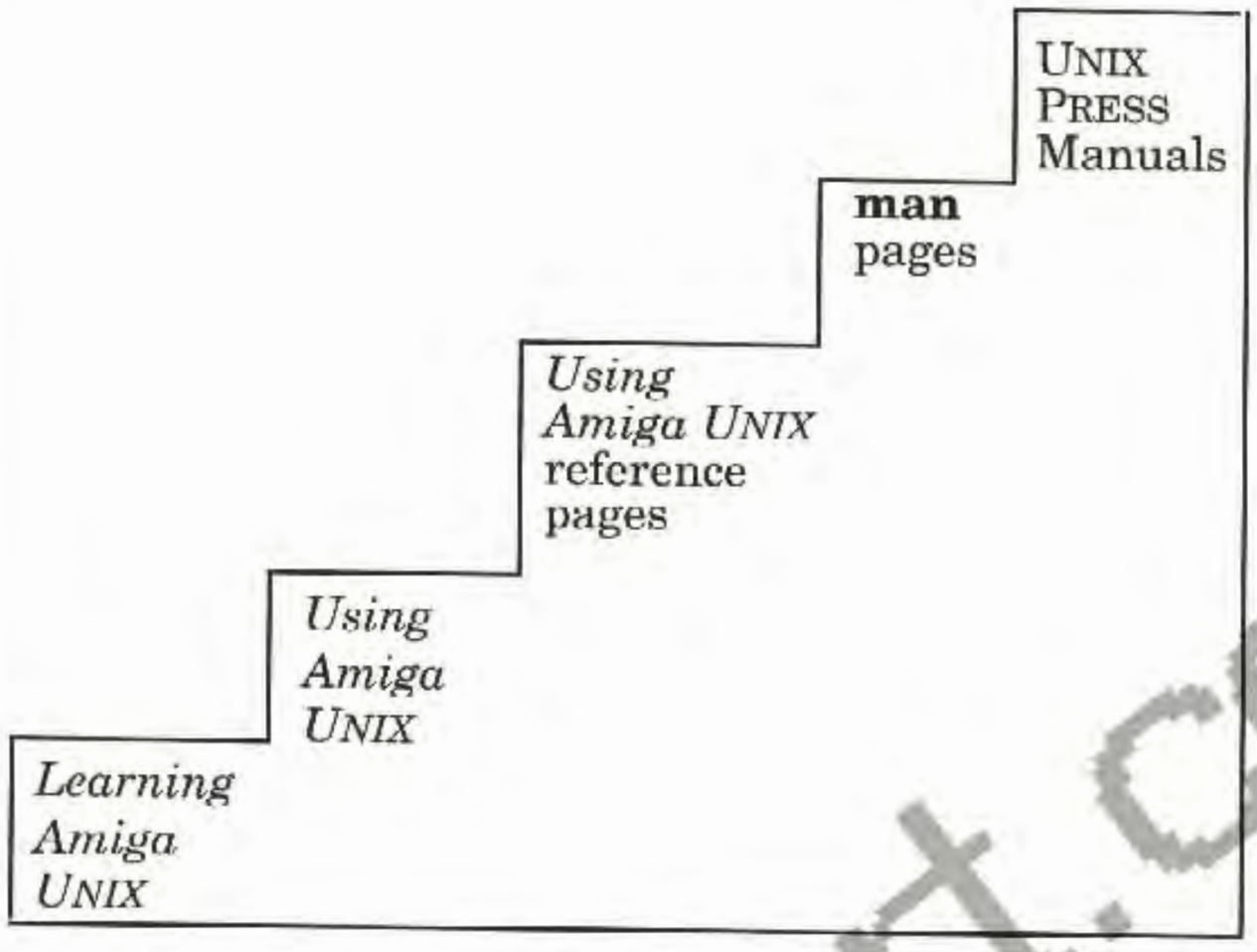
You can also check the reference chapter at the end of the manual; the common MS-DOS commands are listed with their UNIX equivalents and, where appropriate, the combination of procedures you can use to imitate or expand on PC commands.

Notes for new users

New users should read the rest of this introduction; it highlights some important UNIX concepts that you might not have seen before. You should then skim the first few chapters, to see what kinds of commands and tasks are commonly used, then refer back to this book whenever you want to learn more about a specific topic. There is an easy progression to follow to find more information:

Many information sources for new users

- start with the index to this book, then read the appropriate paragraph or page to see how a task is performed
- follow the tutorial in our companion book, *Learning to Use Amiga UNIX*, for hands-on instructions and examples
- read the reference chapter at the end of this book for more options on a specific command
- continue to the online man pages if you still need more information
- read the detailed documentation in AT&T's complete UNIX PRESS documentation set for UNIX System V Release 4



Read reference material in ascending order

As a new user, you can probably become a competent UNIX user simply by reading the first few chapters of this book and following the tutorials in *Learning to Use Amiga UNIX*.

Basic features of UNIX

UNIX has many features that are common on large multi-user computers, some that are unique to UNIX, and almost all of the features normally found in desktop personal computers. Describing these features would take an entire book; in fact, it usually takes about fifteen books. This manual is an overview to some important UNIX features; this page is a brief introduction to that overview.

Note the following features and concepts. You will see them regularly throughout the manual and as you work at your computer using UNIX.

Amiga UNIX virtual screens

- Amiga UNIX adds virtual screens to UNIX; press any ALT-functionkey combination (F1 through F10) to change to a different login screen. Virtual screens make your one Amiga screen work like ten different screens, sharing one keyboard and monitor.

Multiple users

- UNIX is a multi-user and networking system; each virtual screen can be "used" by a different user, as can any attached terminals or any remote network logins. You can communicate with these other users, see what they're doing, and even restrict them from specific files and commands.

If a command does not seem to work . . .

- If a command does not work, you should first try typing it again. Some special characters do not appear on your screen; what looks like a clean command line might have garbage hidden on it. If you type a command several times and it still does not work, check the command syntax in this manual, then your path. It is important to have the command's location in your path; UNIX searches only your path when you type a command, and does not look anywhere else.

CTRL-C to stop a process

- If you start a command, you may not be able to do anything else until it finishes. Some commands can take a long time, particularly if you make a mistake and ask for the wrong operators (listing an entire hard disk, for example, instead of just your directory). Use CTRL-C to stop the current command.

Processes run "invisibly" in background

- UNIX is a multi-tasking system; it runs many processes in the "background", where you cannot see them and they do not visibly interfere with your "foreground" work (what you type and see on the screen). If your system seems slow, you might have large processes running in the background. (UNIX itself always keeps processes running in the background, but it handles these so they do not take much computing power away from you.) To stop a background process, use the **kill** command. To keep working while a long, slow process continues, put it in the background.

Directory tree

- UNIX is very dependent on files and directories; most objects are in files or are processed through files, and all files are in directories. You should understand the basic file and directory concepts and commands documented in the first chapter of this manual.

Online help with man

- UNIX provides extensive online help, although in a somewhat cryptic fashion. Every UNIX command is documented in a "man page"; type **man**, followed by the name of a command, to get the man page for that command. Man pages document all the intricacies of a command, which makes them somewhat difficult for novice users to read and use. It's still good to know that they're there, so you can always look up information to learn about a command or try to fix a problem. (Note that on some smaller systems, a system administrator might remove

the man pages to make room for other files. The man pages represent several books of text and take up a lot of disk space.)

Complete documentation from AT&T

- AT&T provides a complete documentation set for UNIX System V Release 4. Since Amiga UNIX is an exact port of that system, you can read any part of this documentation. It is printed by Prentice Hall, and is available in many bookstores or directly from Prentice Hall. The only features not listed in the AT&T documentation are those which we added to Amiga UNIX and described in this manual.

Getting started

Log in

Type your username at the login prompt. Type your password at the password prompt.

Virtual screens

Alt-*functionkey* combinations control the virtual screens.
Press ALT-F1 through ALT-F10 for different login screens.

OPEN LOOK

oladduser	puts .olsetup in a user's .profile
olinit	start OPEN LOOK

System status

who	Check who is logged into the machine
finger	Similar to who but more details about users
last	List most recent logins for one or more users
ps	List your processes
ps -e	List everyone's processes

UNIX shells

/bin/sh	Bourne shell. Historically the standard AT&T shell.
/bin/csh	C shell. Used by programmers and Berkeley enthusiasts.
/bin/ksh	Korn shell. Compatible with Bourne shell. Some C shell features. Easy command line editing.
/usr/lib/rsh	Restricted shell. Limits a user's capabilities.

Getting Started

Why should you read this chapter?

You should read this chapter if you have never used UNIX before. If you already know UNIX, you should still skim this chapter to learn about the unique screen features of an Amiga terminal.

General features of UNIX

UNIX in general has a few features that distinguish it from personal computer operating systems, including:

- UNIX is multi-tasking
- UNIX allows multiple user sessions
- UNIX has a built-in security system
- UNIX includes networking software and commands

You can have many users and operations all working on your computer at the same time. You should learn how to keep track of these users and processes; remember, a computer that can do many things requires a little extra time to maintain, monitor, and learn.

User accounts

You must have a user account in order to work on a UNIX system. Every user account has a unique name and password associated with it. Unless you know this *username* and password, you will not be allowed on the computer. User accounts, *usernames*, and passwords are UNIX's defense against unwanted users.

Networking

UNIX provides the tools you need to communicate over a network. All you have to do is make the physical connection and configure the network software.

Amiga UNIX virtual screens

Amiga UNIX has a unique feature called virtual screens. You can have up to ten virtual screens, which means you can log in to your single computer ten times. Each of these screens can run any UNIX command, even windowing and graphics commands. It's like having ten different terminals plugged into your computer.

This chapter provides you with the basic information you need to start using Amiga UNIX, to use the virtual screens, and to keep track of users and processes on your computer.

Logging into your computer

Login is a security feature

UNIX provides a security feature called logging in. Any time you want to use a UNIX system, you must first log in by typing your *username* and password.

Type your user name and password

```
login:joe
Password:
```

The password does not appear on the screen

Your password does not appear on the screen as you type it. This prevents other people from reading your password over your shoulder.

All UNIX systems have a user called root. root is the most powerful user on your computer and is sometimes called the "superuser". root has the power to override file protections and user security. root also has the power to damage your system. Log in as root only to perform system maintenance. Create your own user account for your regular work.

You give other users permission to use your computer by setting up a login account for them. See chapter 10, *Maintaining your system*, for information about setting up user accounts.

Change your password

Change your password at any time by using the **passwd** comand.

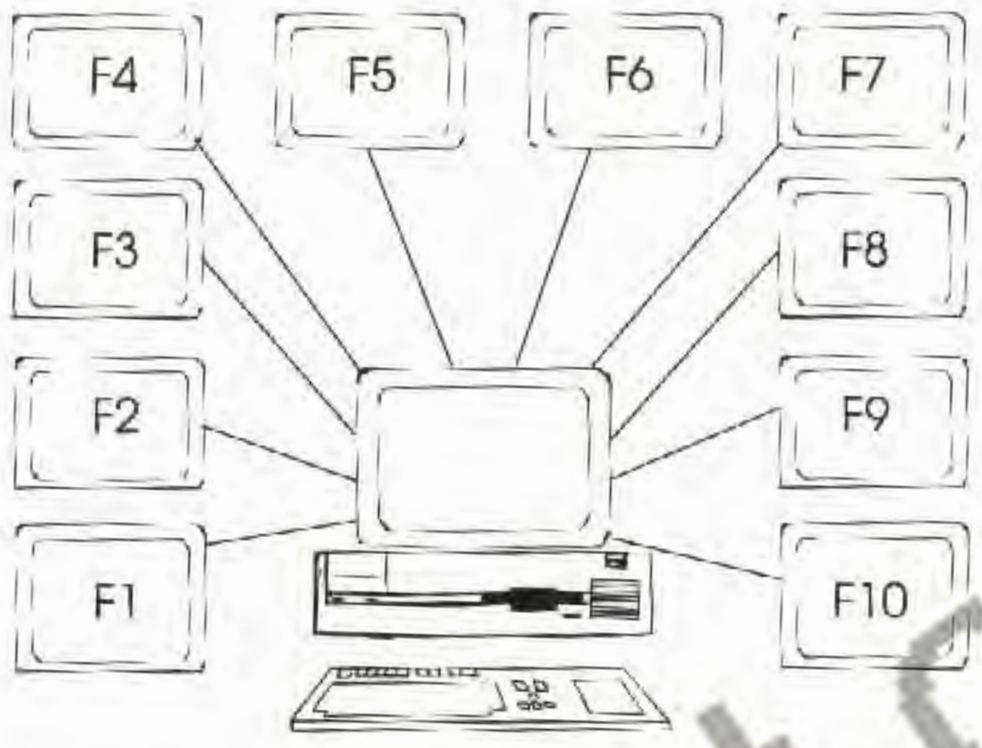
```
passwd:changing password for username
Old password:
New password:
Re-enter new passwd:
```

Type your old password, then your new password, then confirm your new password by typing it again.

Using the virtual screens

Press ALT and a function key to select a screen

Use each virtual screen as if it is a separate terminal. Press ALT-*functionkey* to change to the screen associated with that key.



Press Alt-Function keys for virtual screens

If an empty black screen appears, you have not defined a screen for that function key.

Check each function key for a screen

You can have many processes running on these virtual screens; logging off of one screen has no affect on the others. Check the screens currently running by pressing each ALT-*functionkey* combination in sequence. You should always check the current function keys after logging out, because you might have forgotten about some screens that are still in use.

You create additional virtual screens by executing specific programs from the command line (such as **olinit**, which creates an X screen and an OPEN LOOK session at the maximum resolution of your monitor).

These screens are automatically associated with your current function key; that is, screens you create from F2 are also attached to F2 (stacked one behind the other). Since only the top screen is visible, you have to cancel it to see any other screens behind it.

Change screen characteristics

Screen settings may be different for different screens, since these settings are all customizable by a user or system administrator. By default, eight of the ten function keys are defined, with a variety of different colors, fonts, and resolution levels. You can set display characteristics in any of three ways:

- system defaults
- each virtual screen setting in inittab (see *Editing system files* later in this manual)
- at the command line for a particular session

Change screen color

Change the current color with the **color -bc *color* -fc *color*** command, using **-bc** for background color and a number from 000 to fff. Each position in the number represents an amount for red, green, and blue, respectively. Changing these numbers individually allows a wide variety of color combinations between the darkest, 000, and the lightest, fff. Change the foreground color with **-fc**. In place of the exact number for a color, you can type the standard colors **black**, **blue**, **green**, **cyan**, **red**, **magenta**, **yellow**, and **white**, as shown in the screen on the next page.

set background color to blue

set foreground color to white

```
color -bc blue -fc white
```

Show current screen colors

Look at the current background and foreground colors by typing **color -show**. Many colors are listed; the first two are the background and foreground colors, respectively.

background color

foreground color

```
c00 000 c01 fff c02 0fe c03 7f7 c04 03b c05 0cf c06 004 c07 0b6
c08 000 c09 000 c10 000 c11 000 c12 097 c13 0a4 c14 0fe c15 00c
c16 097 c17 02b c18 089 c19 000 c20 7fb c21 0f0 c22 004 c23 000
c24 005 c25 7fb c26 005 c27 07f c28 032 c29 087 c30 089 c31 007
```

Choose a font type and size

Because of the Amiga's unique graphic processor, you can use different fonts, sizes, and colors, even when you are not running a graphic program (such as X) or graphical user interface (such as OPEN LOOK). You can get a login prompt, request and see UNIX commands, and even edit files in a variety of Amiga fonts.

These screens are automatically associated with your current function key; that is, screens you create from F2 are also attached to F2 (stacked one behind the other). Since only the top screen is visible, you have to cancel it to see any other screens behind it.

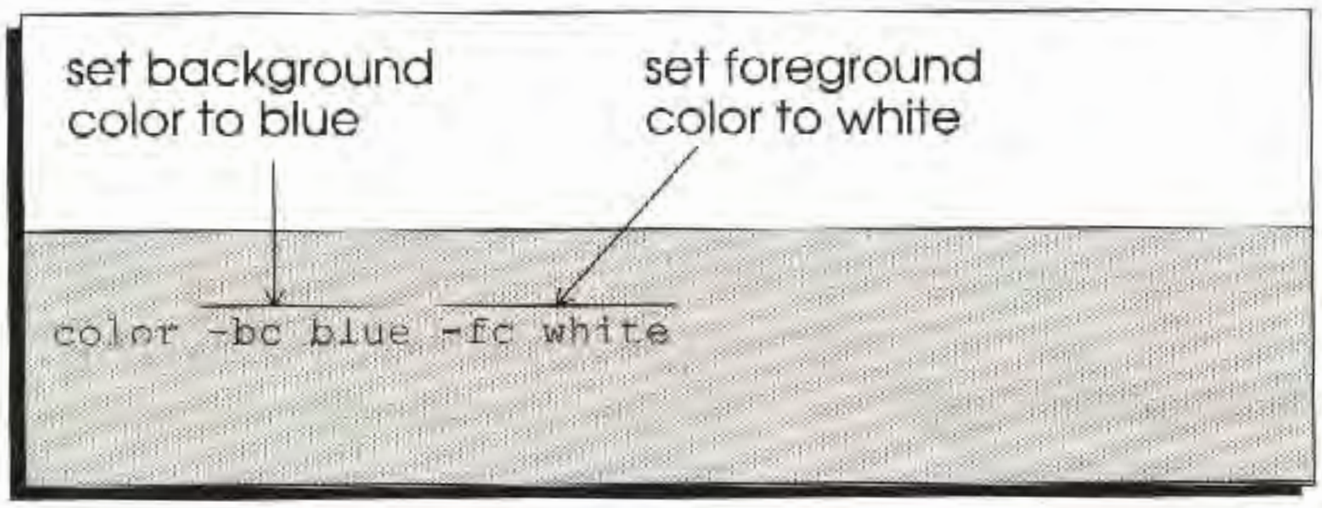
Change screen characteristics

Screen settings may be different for different screens, since these settings are all customizable by a user or system administrator. By default, eight of the ten function keys are defined, with a variety of different colors, fonts, and resolution levels. You can set display characteristics in any of three ways:

- system defaults
- each virtual screen setting in inittab (see *Editing system files* later in this manual)
- at the command line for a particular session

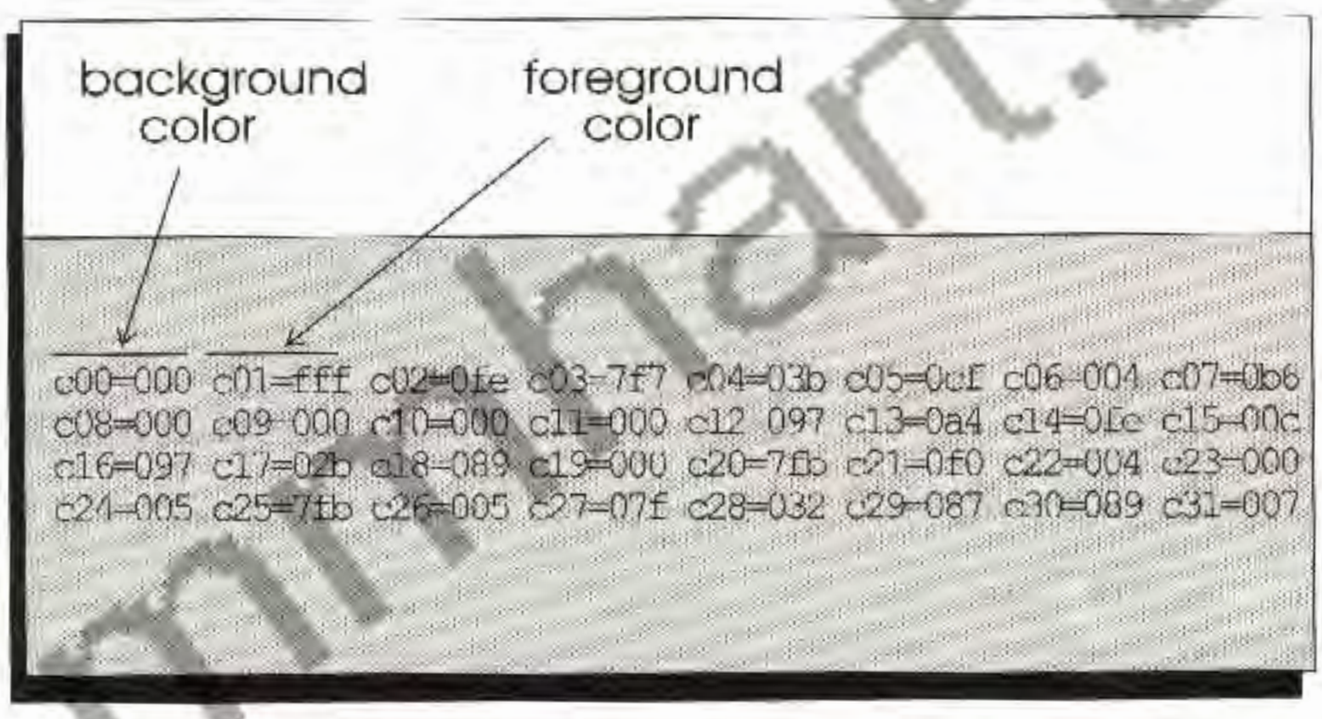
Change screen color

Change the current color with the **color -bc *color* -fc *color*** command, using **-bc** for background color and a number from 000 to fff. Each position in the number represents an amount for red, green, and blue, respectively. Changing these numbers individually allows a wide variety of color combinations between the darkest, 000, and the lightest, fff. Change the foreground color with **-fc**. In place of the exact number for a color, you can type the standard colors **black**, **blue**, **green**, **cyan**, **red**, **magenta**, **yellow**, and **white**, as shown in the screen on the next page.



Show current screen colors

Look at the current background and foreground colors by typing **color -show**. Many colors are listed; the first two are the background and foreground colors, respectively.



Choose a font type and size

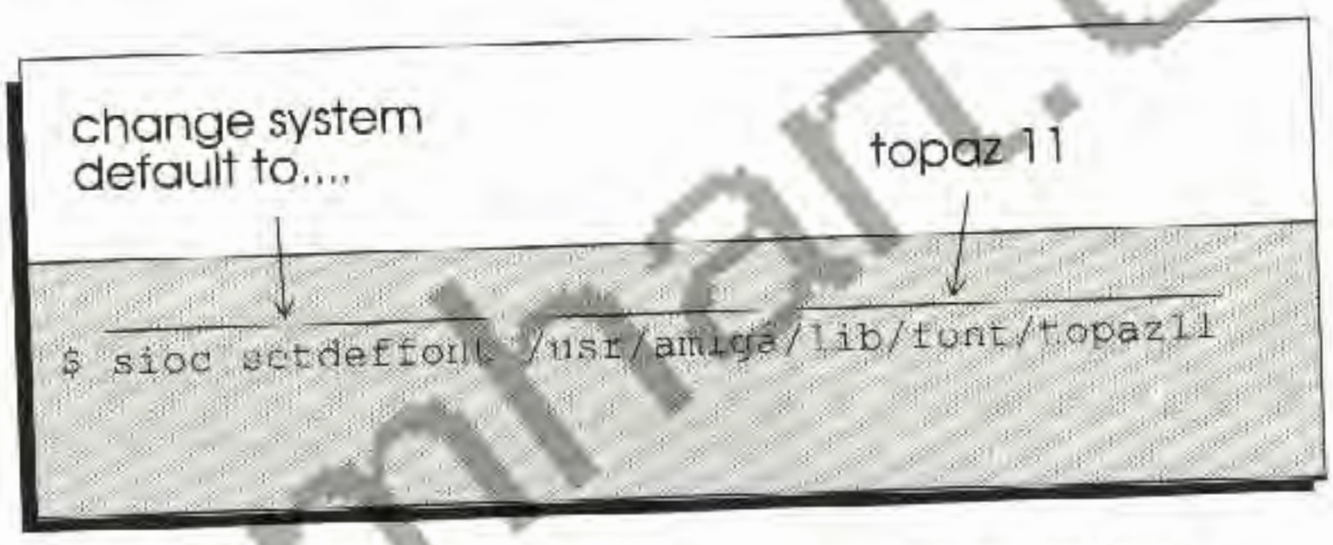
Because of the Amiga's unique graphic processor, you can use different fonts, sizes, and colors, even when you are not running a graphic program (such as X) or graphical user interface (such as OPEN LOOK). You can get a login prompt, request and see UNIX commands, and even edit files in a variety of Amiga fonts.

At the present time, two Amiga fonts are shipped with Amiga UNIX release:

- topaz8 (the default)
- topaz11

Change the font size and type

Use **sioc** to change either the current font or the default font. **sioc setfont *filename*** changes the current font for this login session on this screen; **sioc setfont** resets the screen font to the default system font; and **sioc setdeffont *filename*** changes the system default font.



After they start, a short menu appears. Select "Programs..." from this menu. Another menu appears with only one option: "Xterm..."; click on this option to start an X terminal window.

Using OPEN LOOK

What is OPEN LOOK?

OPEN LOOK is a graphical user interface that lets you run terminal windows and programs by clicking menu options and icons. You're still working in UNIX, but instead of having to type all the commands, you can select them from pull-down menus, operate others with a mouse, and have several running at the same time in different windows.

Setting up OPEN LOOK

OPEN LOOK doesn't start by default, you configure it to run whenever you want. You only have to set up OPEN LOOK once, after that you can start it by typing a single command.

To create your own OPEN LOOK environment, you first have to run the **oladduser** command. This program puts various OPEN LOOK and X Window System resource files in your home directory, and modifies your .profile.

```
$  /usr/X/adm/oladduser username
$  CTRL-D
```

Log out then log back in to use your changed startup file.

OPEN LOOK and the C shell

The **oladduser** command doesn't work with the C shell. If you use the C shell, copy the following files from /usr/X/adm into your home directory:

- .Xdefaults
- .olinitrc
- .olprograms

You also need to add the following lines to your .login file:

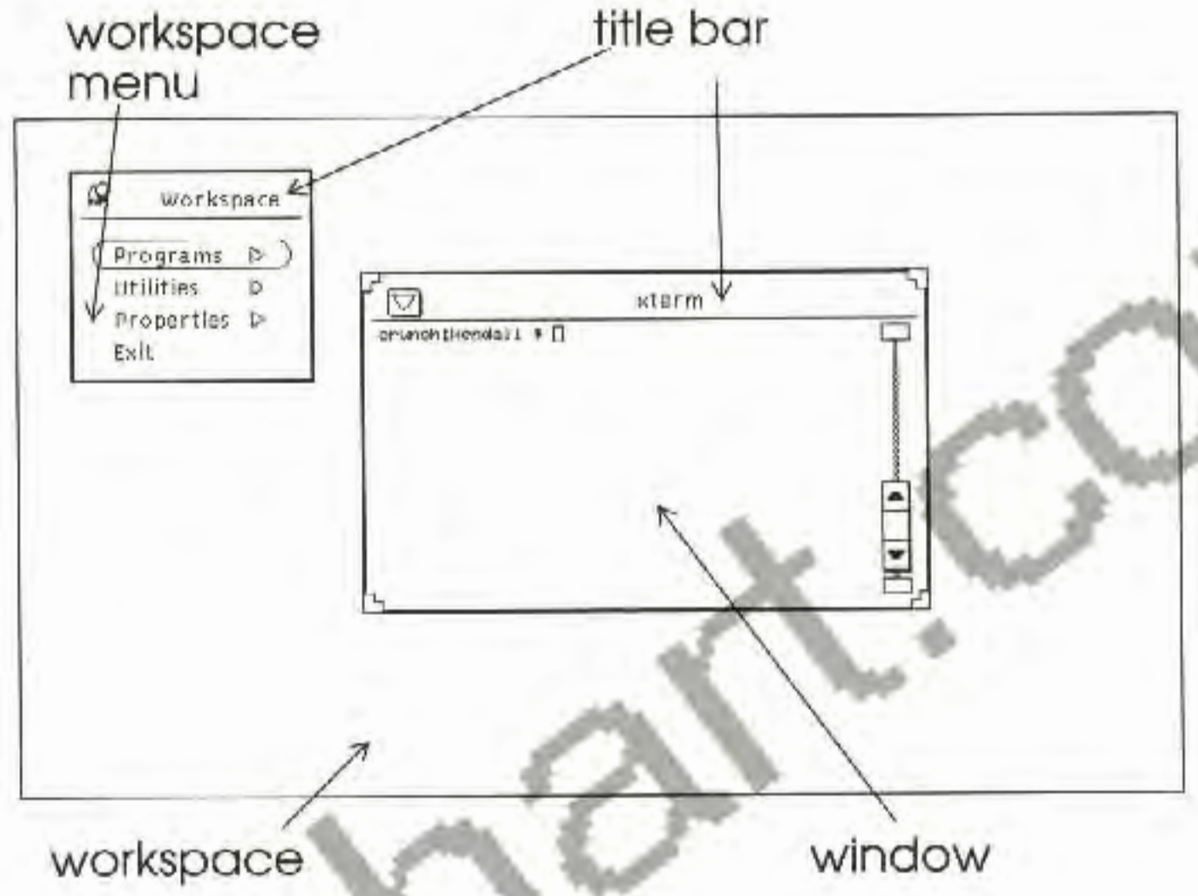
- setenv DISPLAY unix:0
- setenv XNETACCESS on
- set path=($path /usr/X/bin)

Start OPEN LOOK

To start OPEN LOOK, type **olinit** and wait a minute for the OPEN LOOK windows and X Window System server to start.

The OPEN LOOK workspace

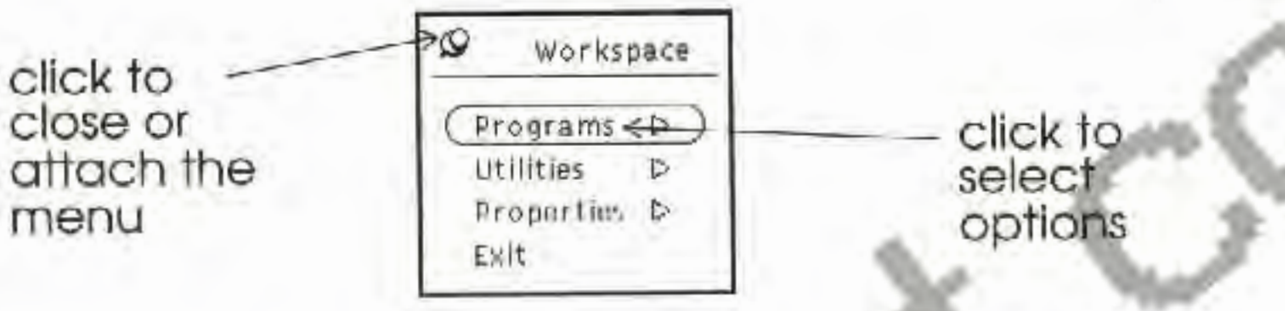
After OPEN LOOK starts, you see a single menu on a blank screen. The following figure shows the workspace with an OPEN LOOK window and the workspace menu.



Menus

Most menus under OPEN LOOK work the same way:

- make them appear (if not already visible) by clicking the right mouse button
- close or attach them by clicking the push pin at the top left corner of the menu with the left button
- select options by clicking on them with the left button.



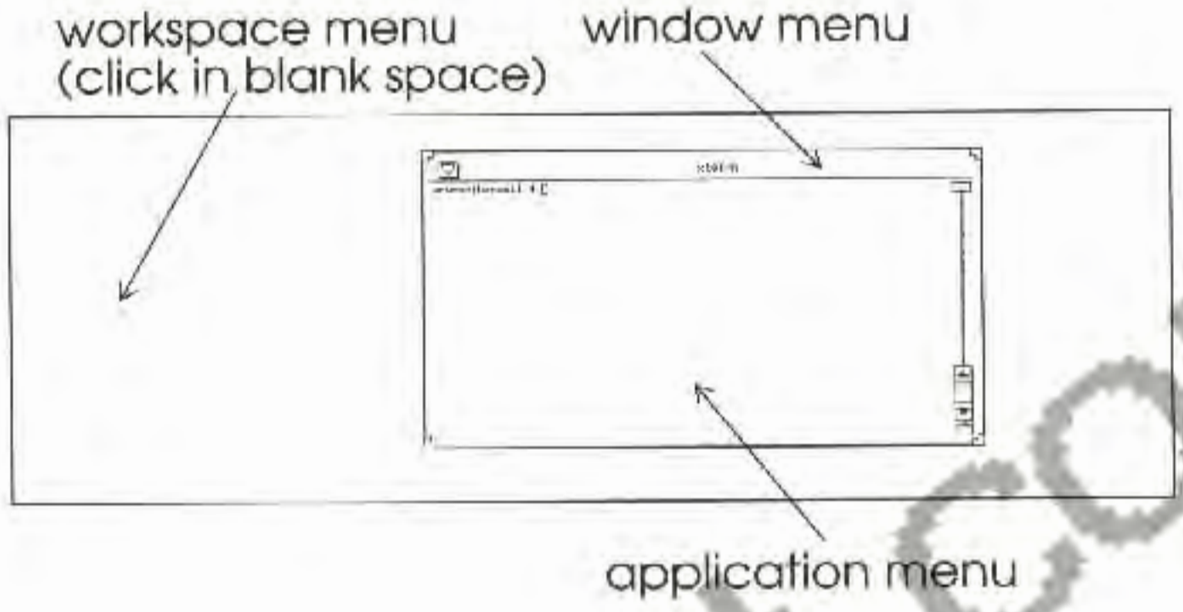
The mouse buttons are similarly consistent: the right mouse button usually brings up a menu and selects options from it, and the left button usually selects or drags an object.

Three primary types of menus

The three primary types of OPEN LOOK menus are: workspace, window, and application. Each type of menu is position-specific; you get a different one depending on where the mouse pointer is when you click the right mouse button.

If you have an OPEN LOOK window open, move the mouse pointer into it and click the right mouse button. A menu for that window appears. Do the same thing for a different application and a different, application-

specific, set of menu options appears. You can also click on the blank workspace to get the workspace menu, or in the title bar of a window, to get the window menu.



Three types of menu options

OPEN LOOK menu options do one of three things:

- start a submenu (...)
- start a dialogue box (>)
- start a program (no marker)

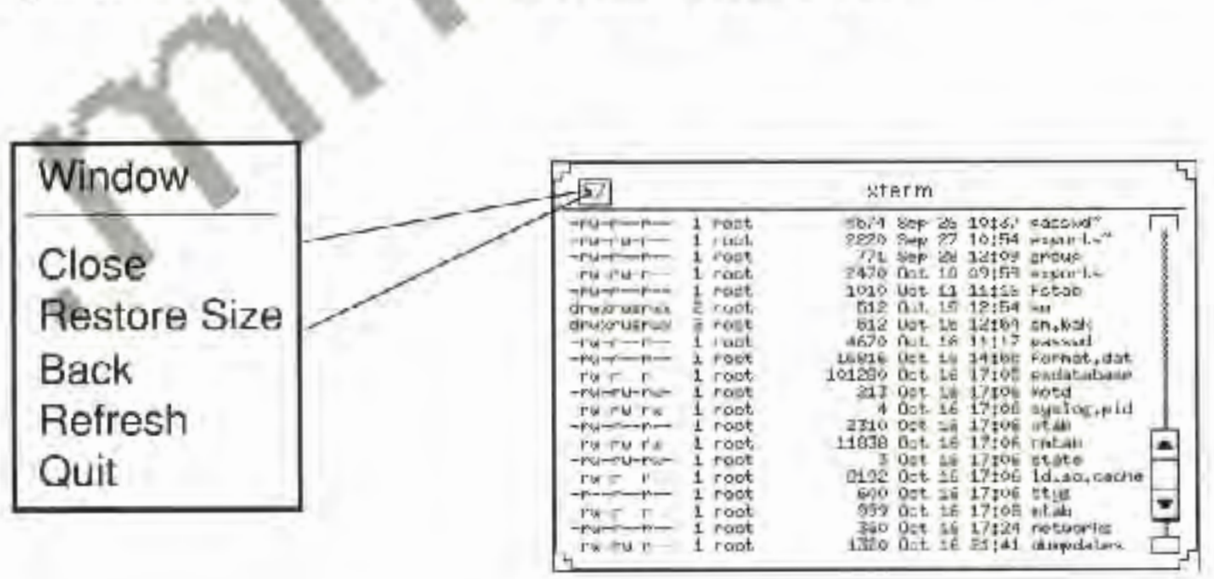
The workspace menu

The workspace menu is the first menu you see after you start OPEN LOOK. You start all other OPEN LOOK programs, menus, and windows using the workspace menu. The table on the following page summarizes the workspace menu options.

Option	Where does it lead?
Programs	X programs and terminal windows
Utilities	Programs for managing the workspace
Properties	Change settings for mouse, icons, or keyboard
Exit	Quit OPEN LOOK completely and return to regular command prompt

The window menu

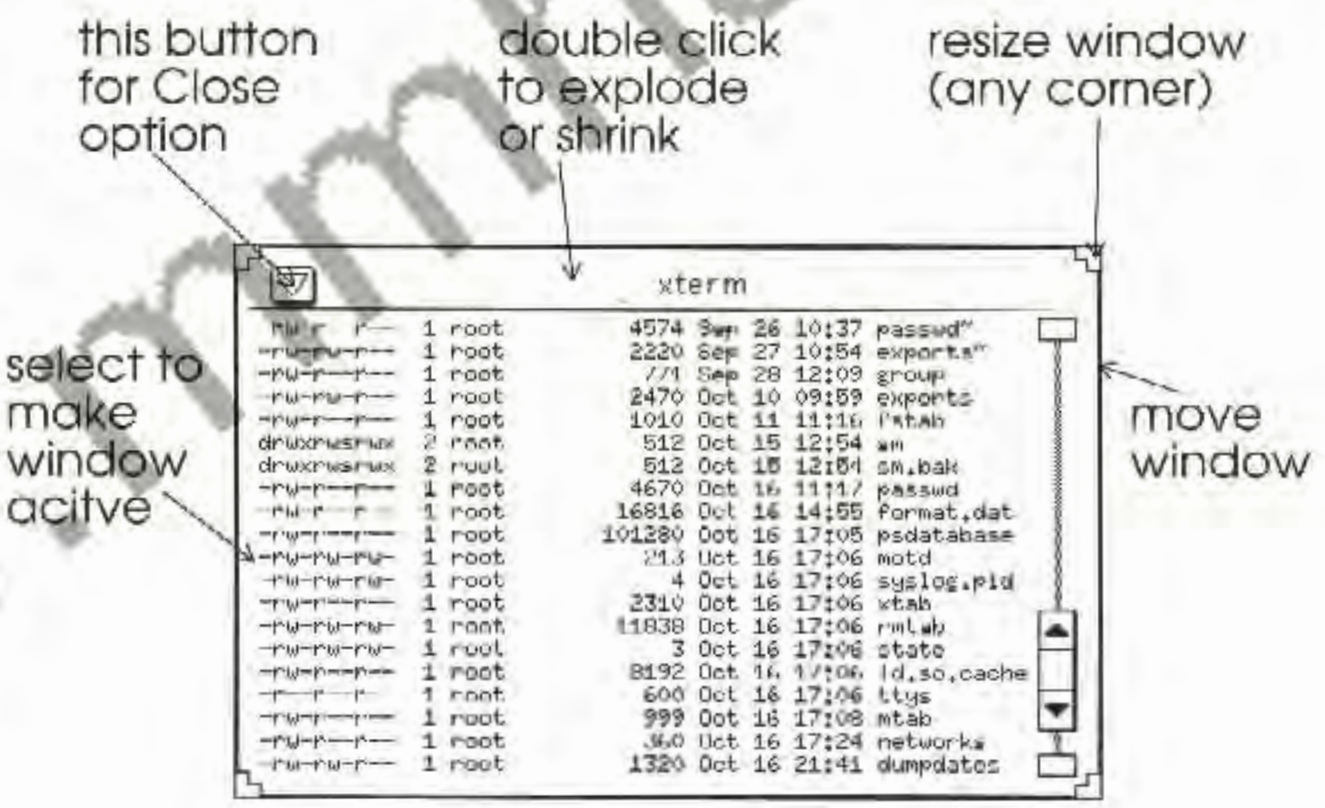
Another special menu, the window menu, has options for manipulating the window. The window menu options are the same for all windows.



Some common features of OPEN LOOK windows

OPEN LOOK programs run in windows. Most OPEN LOOK windows share the following features:

- Resize a window by clicking on one of the four corners and dragging.
- Move a window by clicking anywhere on the border
- Explode to fill a screen by double clicking on the title bar. Double-click again to switch back to normal size.
- Shrink a window by selecting **Close** from the window menu.
- Make a window active by clicking the left mouse button just inside the border or on the title bar. The title bar on the active window is black. Any characters you type appear only in the active window, regardless of where the pointer is.
- Scroll through a window by clicking the scroll bar on the right side of the window.



Active windows

You can open many windows at once, but only one can be active at a time. You can tell which window is active because it is usually the one at the very front (on top of the others) and its title bar is filled, while the others are empty.

The terminal window

The terminal window (xterm) is OPEN LOOK's equivalent of a virtual screen. You can type any UNIX command in a terminal window. The main difference is that virtual screens let you run foreground processes on different screens (function keys), while OPEN LOOK lets you run foreground processes in different windows on the same screen.

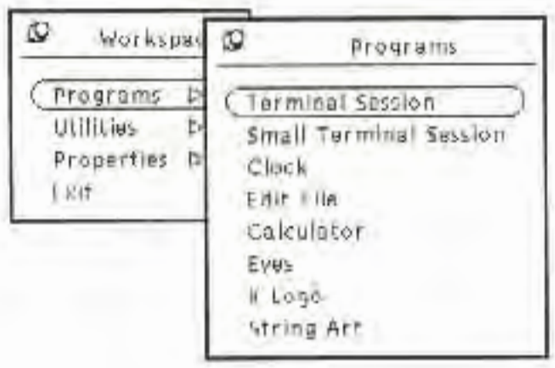
Advantages of a terminal window

Terminal windows have many advantages:

- you can cut and paste commands between windows
- you can scroll backwards to look at old commands
- you can open many windows at once
- you can change the window's size
- you can shrink the window so it is temporarily out of the way, even while a process is active

Start a terminal window

Start a terminal window by selecting **Programs** from the Workspace menu, then **Terminal Session** (large font) or **Small Terminal Session** (small font) from the Programs menu.



Cut and paste between terminal windows

You can move text between xterm windows by cutting and pasting. You copy text from one xterm (source), and paste it into the other xterm (destination).

To copy text, you must first select the text you want to copy. Make the source xterm active by clicking on its title bar. Select the text you want to copy by putting the mouse pointer at the beggining of the text, clicking the left mouse button, and dragging unitl the text is darkened.

source xterm with text selected

destination xterm where you are going to paste the text

Copy the text

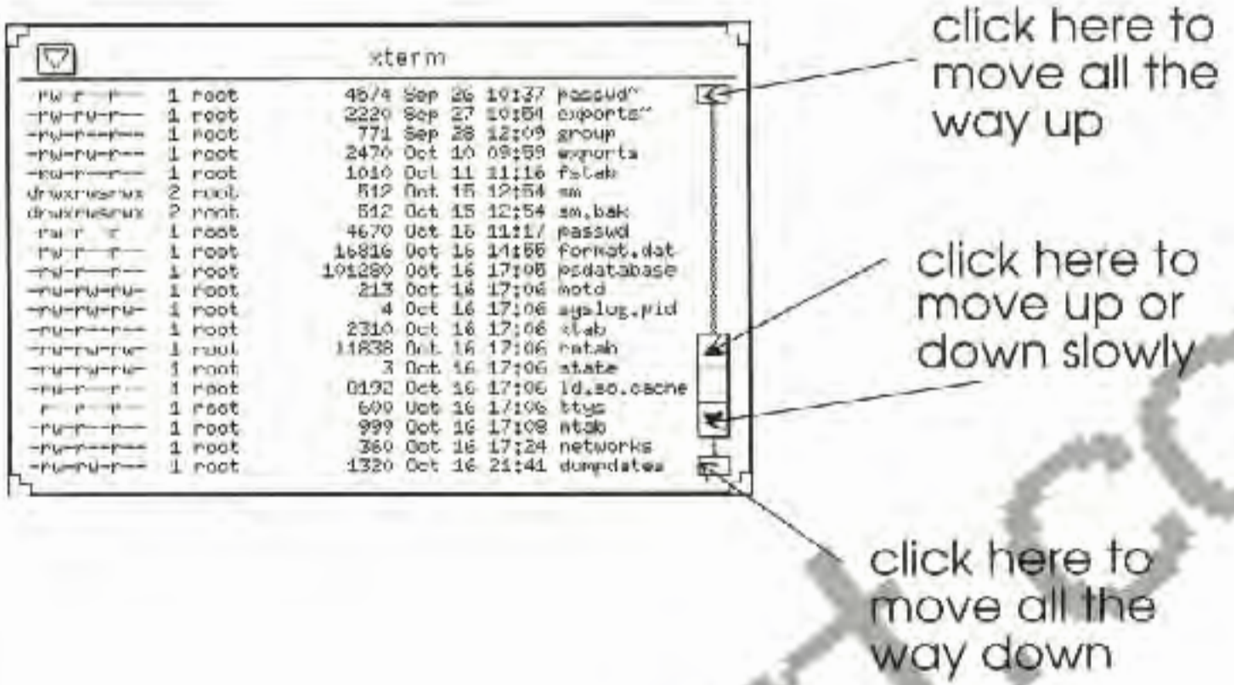
Copy the text using one of two methods: get the xterm menu (by pointing anywhere in the xterm window and clicking the right mouse button) and select Edit, then Copy from the Edit menu. As an alternative to the Edit menu, you can press CTRL and F2 to copy the text.

Paste the text

Paste the text into the destination xterm by making that xterm active, positioning the mouse pointer where you want the text to appear, and pressing CTRL F4. You can also use the select the xterm menu, then Edit, then paste from the Edit menu to paste text.

Scrolling back through old xterm commands

As you type more and more commands into the xterm window, the old ones scroll off the top of your screen. You can scroll back through them by using the scroll bar on the right side of the window.



Customizing your environment

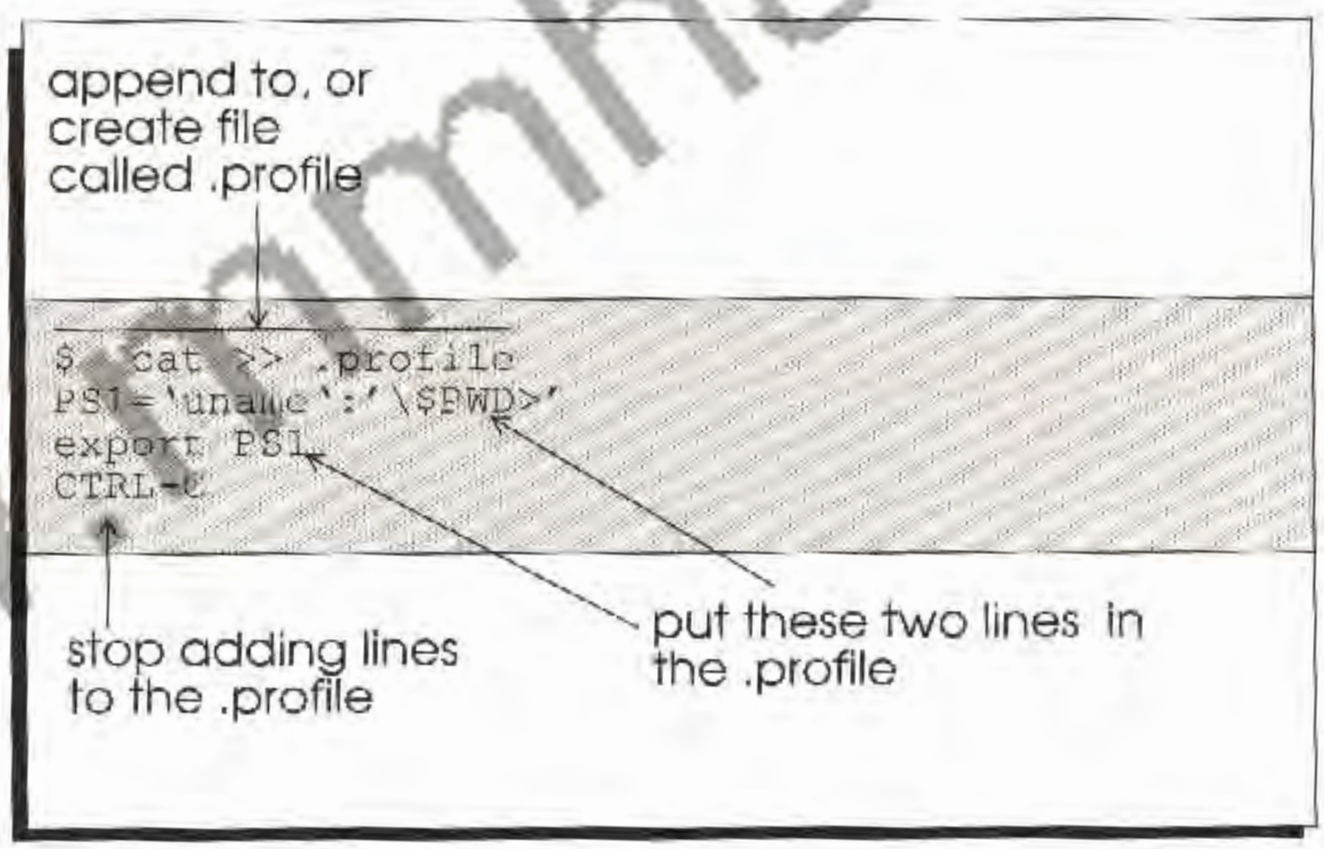
Customizing your login environment

When you first start UNIX, you are working in the default environment we set for you. You can change this environment by editing (or creating) a file called .profile in your home directory and filling it with your own custom settings.

Your shell looks for a startup file in your home directory that contains information specific to you (such as what your prompt should look like and whether you want OPEN LOOK to start automatically). The Bourne and Korn shells (see *Understanding the UNIX shells* later in this chapter) use .profile, and the C shell uses .login. Both serve the same purpose.

As an example of how a startup file affects your login session when you are using the Korn shell, type the following lines to put a file called .profile in your home directory.

Making a custom ksh prompt



Log out and log in again. Your prompt should have the output of uname and the variable PWD, to show your system name and current directory.

```
amiga:/home/username>
```

Set up aliases

You can assign an alias to practically any UNIX command. Aliases let you change the name of a command, set the options for that command, or even combine commands. In fact, aliases let you completely customize UNIX commands to suit your needs.

You define aliases in your .profile for the Korn shell (or .login for C shell). For example, you could make a new command to display the long ls sorted by time (**ls -lt**). Here's what you enter to make **lt** display **ls -lt** by default.

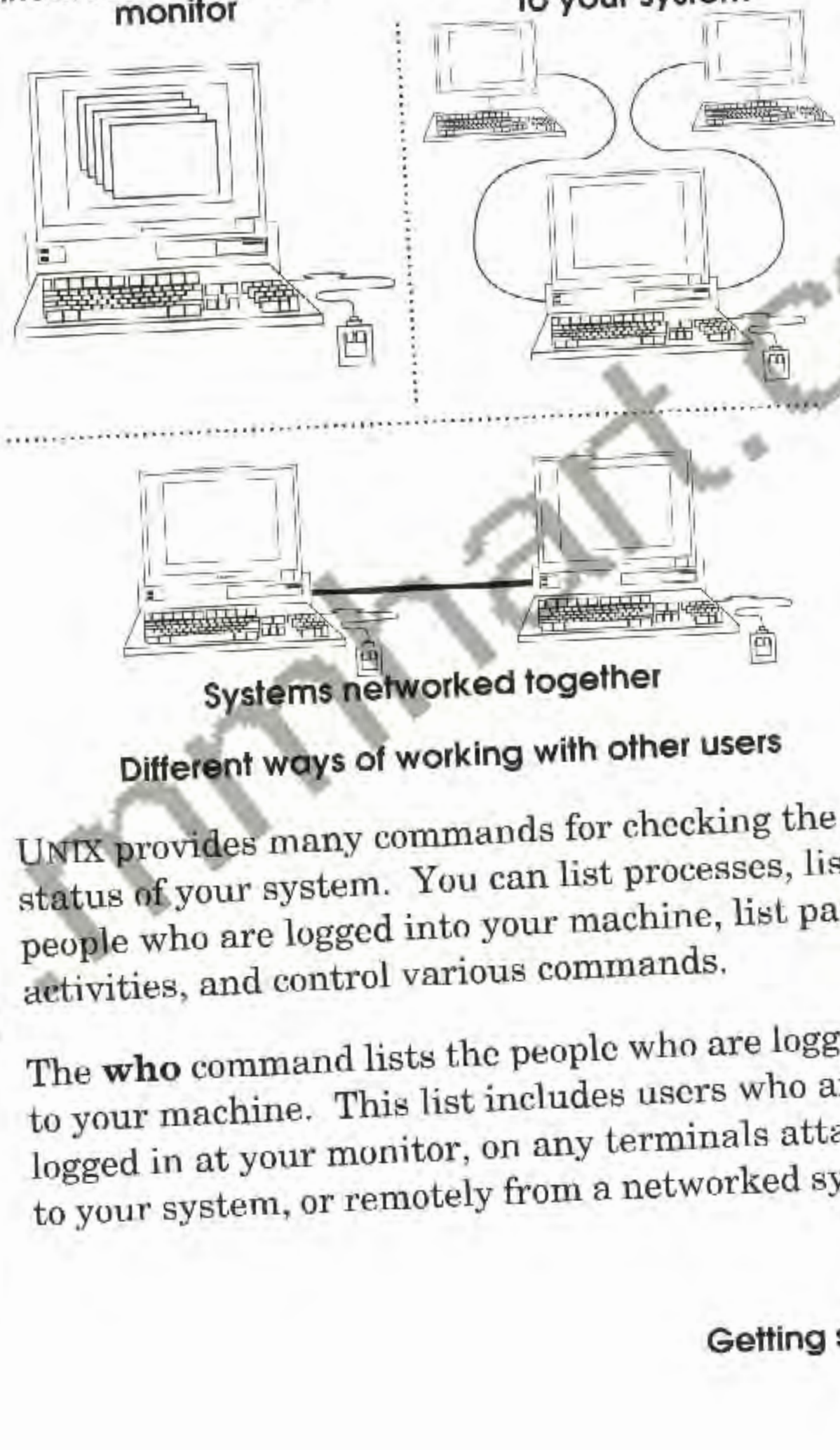
```
alias lt='ls -lt'
```

NOTE: The format for the **alias** command depends on the UNIX shell you are using. (See *Understanding the UNIX shells* later in this chapter.)

Checking on users and operations

UNIX is a multi-user operating system

UNIX is a multi-user, multi-tasking operating system. You can have many people and operations working on your computer all at once.

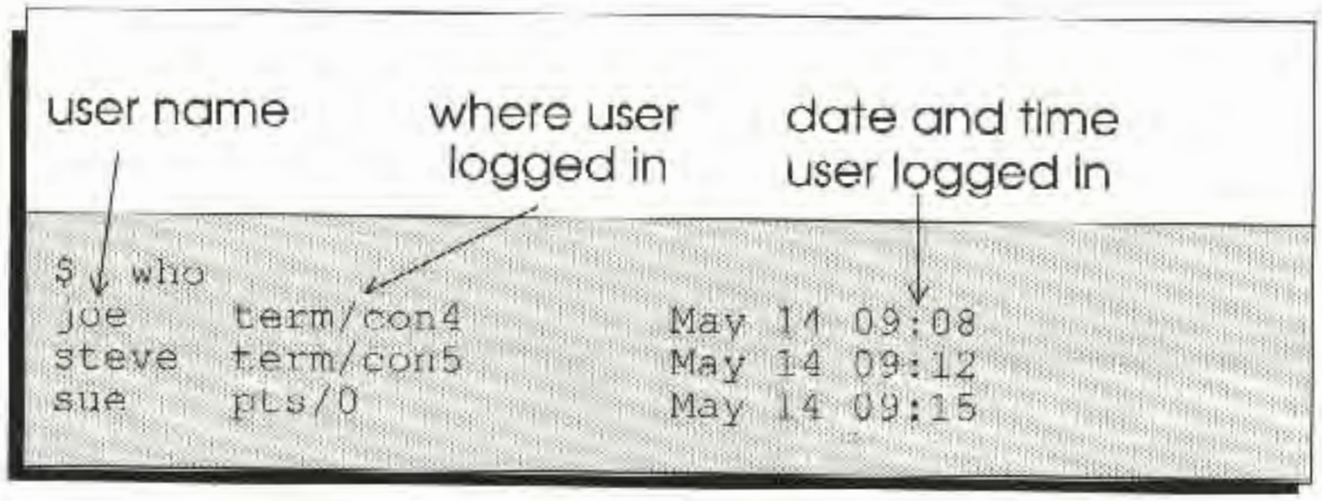


Different ways of working with other users

UNIX provides many commands for checking the status of your system. You can list processes, list people who are logged into your machine, list past activities, and control various commands.

Check to see who is logged in

The **who** command lists the people who are logged in to your machine. This list includes users who are logged in at your monitor, on any terminals attached to your system, or remotely from a networked system.



This example tells you that there are three people using your computer: two of them through the virtual screens on your monitor (term/con4 and term/con5) and one from over the network (pts/0). Remember that all the virtual screens share your monitor and keyboard; so even though two login sessions may be active, only one person is actually sitting in your chair using your keyboard.

Check on users with finger

Another useful command for checking on other users is **finger**. **finger** is similar to **who**, except it provides a little more information. Here's an example showing the output from both **who** and **finger**:

Comparing finger to who

```
$  finger
Login Name       TtY         Idle When         Where
joe   Joe Jones term/con4 12    May14 04:08 amiga

$  who
joe              term/con4      May14 04:08
```

Use options with finger

The most useful finger option is probably **-l**.

```
$  finger -l
Login Name:joe                         In real life: Joe Jones
Directory: /home/joe                   shell/bin/ksh
On since Aug 27 09:05:46 on term/con4
1 hour Idle Time
You have mail in /var/mail/joe
```

In addition to the standard **finger** information, the **-l** option lists the user's shell, time they have been idle, and status of their mailbox.

Use finger on one user

You can also specify one user with **finger**.

```
$  finger joe
```

This displays login information only for joe, if he's logged in. It's a little faster than **finger**, since you don't have to read through many lines of information for other users.

Check on running processes

The **who** command only tells you the names of the users logged into your system. It doesn't tell you what these people are doing. One of your routine system administration tasks should include running the **ps** command. **ps** gives you a list of your processes.

```
$ ps
PID      TTY      TIME      COMMAND
1234     ttyp4    0:01      ps
1237     ttyp1    0:01      ksh
```

This tells you that you have two processes on your machine: the **ps** command (which you just typed), and **ksh** (the Korn shell).

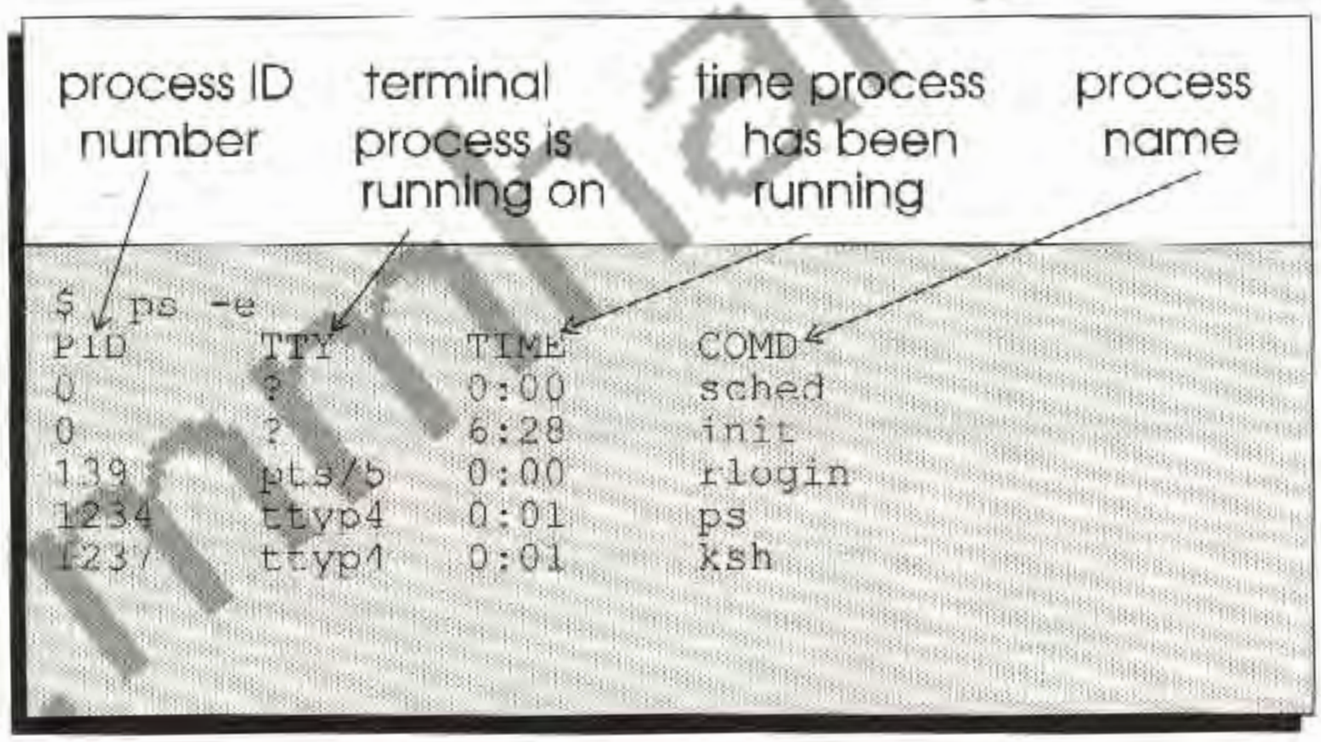
ps has many options. The one you are most interested in is **ps -e**. This command gives you a list of all processes on your system, including yours, other users', and system processes that are always in the background.

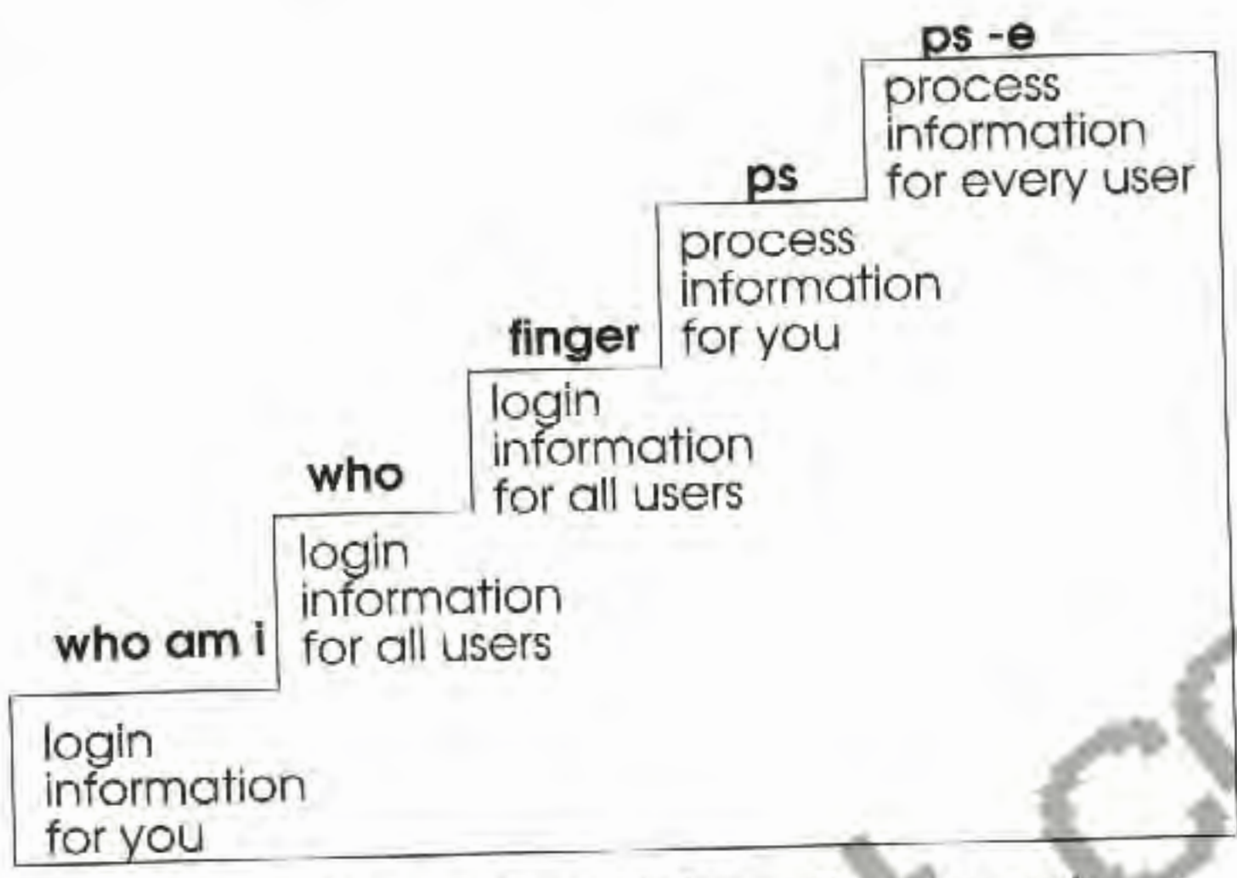
Use more with ps

If there is a large list of processes, they scroll by without stopping. You can use the **more** command to look at one screen at a time:

```
$ ps -e | more
```

This says pipe (the vertical bar) the **ps** command's output through the **more** program so you can look at one screen at a time. See *Working with files and directories* in this book for help using the **more** command.

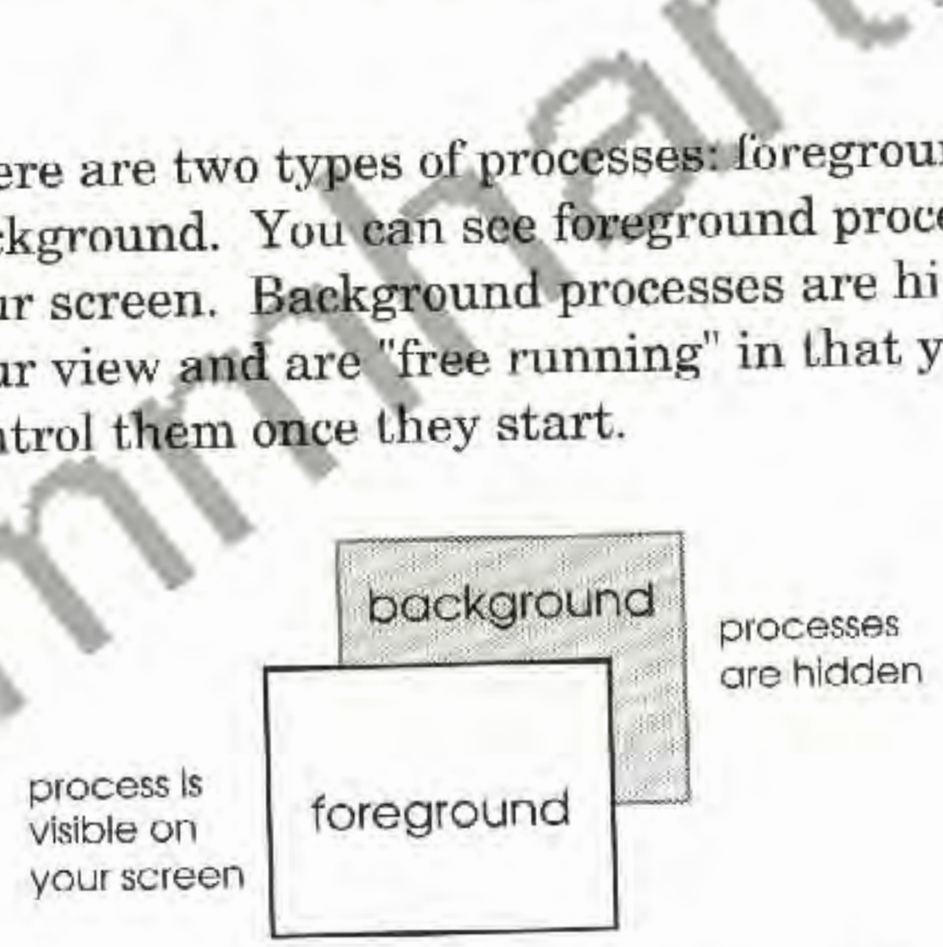




The progression of information requests

Foreground and background processes

There are two types of processes: foreground and background. You can see foreground processes on your screen. Background processes are hidden from your view and are "free running" in that you don't control them once they start.



Foreground and background processes

Kill a process using the ID number

Foreground processes can usually be stopped with CTRL-C; background processes and some foreground ones can only be stopped by a kill command.

One very important piece of information from **ps** is the process ID (PID) number. This unique number identifies a process. You can use the process ID number to terminate a process with the kill command. You might need to kill a process if it freezes, isn't working properly, or if CTRL-C doesn't work.

Use the **ps** command to find the PID (look for the process name first in the **ps** list) then **kill** to cancel it. If your terminal is frozen, press ALT-*functionkey* to switch screens, then log in, **ps** and **kill** the bad process from there.

Use the kill command

The basic **kill** command works like this:

```
$ kill PID
```

NOTE: The **kill** command is as dangerous as it sounds. If you accidentally kill the wrong process, it's gone. Use **kill** with extreme caution.

You can only kill your own processes. If you want to kill someone else's process, you must first log in as root. As root, you can kill any process you find running on your machine.

Sometimes you can't kill a process by typing **kill** ***PID*** by itself. This usually happens if the process you are trying to kill is hung, or is a system process. If you are having trouble killing a process, type the following **kill** command:

```
$ kill -9 PID
```

The -9 guarantees that the process will die.

Checking on disk usage

Check on disk usage for a directory

You should keep tabs on the amount of disk space you and other people use. If you notice that your hard disk is filling up rapidly, you should check the space used by each person. You do this with the **du** (disk usage) command. It shows you how space is used in a directory branch, including the total for that directory (in 512 byte blocks) and all its subdirectories.

```
$  du /home/joe
764   /home/test
756   /home/joe/mail
5052  .
```

Check on total disk usage for a directory

This example shows disk usage for Joe's home directory (/home/joe). The numbers before each directory name are the number of blocks each directory uses. A block is equal to 512 bytes. You can look at only the total for this directory by typing **du -s**, or the size for all files with **du -a**.

```
$  du -a /home/joe
2.    /home/joe/.rhosts
2.    /home/joe/.sh_history
20.   /home/joe/test
764   /home/joe/test
756   /home/joe/mail
5052  .
```

According to **du**, joe is using 5052 blocks of disk space, or 2.5 Mb. The following table shows how blocks, kilobytes, and megabytes relate.

Blocks	Exact size	Approximate size
1	0.5 KB	
2	1.0 KB	
1,000	500 KB	0.5 MB
2,000	1,000 KB	1 MB
200,000	102,400,000 bytes	100 MB

Use df to check the status of a disk

You can also use the **df** command to check the status of your disks. The **-k** option with **df** lists the total space available on your disks, amount of space used, amount of space left, percentage of disk used, and the mount point.

```
$ df -k
Filesystem          kbytes    used   avail   capacity    mounted on
/dev/dsk/c6d0s1     90112     65256  24856   72%         /
```

Checking on print jobs

Use lpstat to check on print jobs

You can check the status of every job waiting to print on your computer using **lpstat -o**.

```
$ lpstat -o
Rank     Owner     job   Files  Total Size
active   joe       145   mywork 12456 bytes
```

lpstat without any options shows only your jobs waiting to print on the default printer.

Check the printing chapter later in this book for more information on printing.

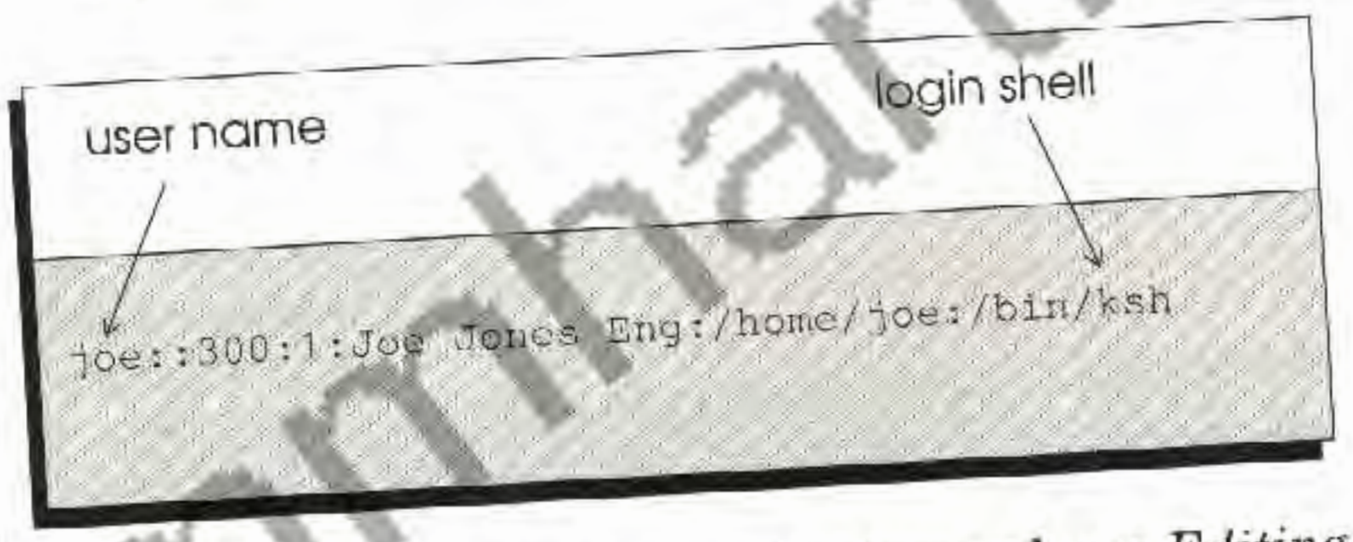
Understanding the UNIX shells

What is a shell?

The shell interprets what you type and acts on it, generally by finding and running a program.

Your default shell is defined in a file called /etc/passwd. Each time you log in, the login command reads your line in /etc/passwd and starts the shell for you. You can also change shells temporarily, after you log in, by typing the shell's name (**sh**, **csh**, or **ksh**). CTRL-D ends your current shell, either going back to your login shell or ending a login session altogether.

The /etc/passwd file contains information about users who are allowed to login to your computer, including their login shell. Here is a sample line from /etc/passwd:



For more information about /etc/passwd, see *Editing system files*, later in this book.

What are the different shells?

You can choose from 5 shells.

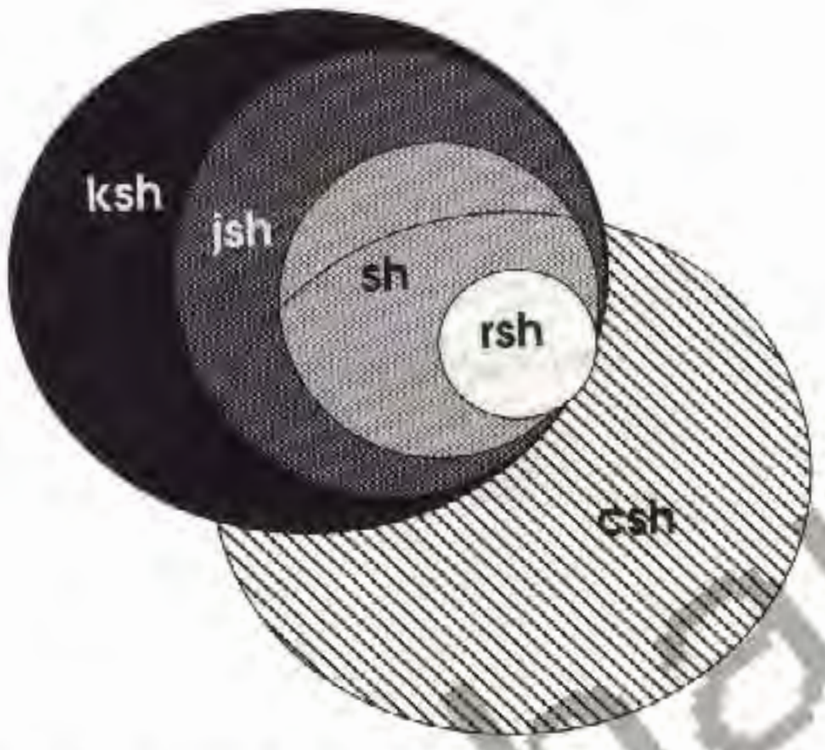
Shell	Description
/bin/sh	historically, the standard AT&T (Bourne) shell
/bin/csh	C shell commonly used by programmers and Berkeley enthusiasts
/bin/ksh	Korn shell, compatible with the Bourne shell, includes many C shell features
/usr/lib/rsh	restricted shell, ignores most commands
/bin/jsh	adds job control functions to sh

What's the difference between the shells?

The Bourne shell (sh) is the standard shell; most other shells were developed from it: rsh is a restriced subset of sh commands, jsh adds job control functions to sh commands, and ksh contains the commands of sh and jsh, plus many other features. The Korn shell (ksh) is the default Amiga UNIX shell and the one we recommend.

The C shell (csh) is a different type of shell; it's based on the Berkeley UNIX system and is not compatible with sh. However, csh does use some of the same commands as ksh.

csh and ksh use different startup files and syntax for certain commands. It's important that you follow the directions for your shell, particularly when doing, comand line editing, using the alias command, or setting shell variables.



How shells are related: csh includes some features from all the standard shells

What's best shell?

We recommend the Korn shell because it is compatible with all Bourne shell functions and provides some C shell features. Some of its convenient enhancements include aliases, command history, command line editing, and job control. See the reference section on login shells in the *UNIX command reference* chapter for more information.

Restricting user accounts with /bin/rsh

You can use the restricted shell (rsh) to limit the things a user can do on your system. The restricted shell prevents a user from:

- changing out of his home directory (thereby getting files other than his own)
- changing the PATH variable (thereby getting at commands that have been hidden from him)
- using a command that contains a / (slash)
- redirecting input and output

It takes more than just rsh

The restricted shell alone isn't enough to prevent users from getting around on your computer. In fact, it's quite simple to break out of a restricted shell unless you set up a complete restricted environment. This environment should include:

- a special bin directory with limited commands (ones that don't allow you to issue shell commands that override rsh)
- a special .profile and PATH for restricted users
- an editor that does not let you execute shell commands

If you don't set up the restricted path and command directory appropriately, an **rsh** user could simply type a standard command and use its features to break your security.

The restricted shell, properly implemented, is enough to contain all but the most experienced users, so you can let students, beginners, and children play with your computer without causing any damage.

However, it is extremely difficult to devise security mechanisms that are completely foolproof against UNIX experts.

Getting help from the man pages

Using the man pages

The Amiga UNIX **man** pages are an online reference manual. The **man** pages are stored in a number of directories under /usr/man. You can check these directories for a list of **man** page topics. After formatting a **man** page, the **man** page is saved in /usr/catman, so it displays faster the next time you request it.

Man page commands

man *topic*	Displays **man** pages for *topic*.
man man	Displays a **man** page that describes how to use the **man** pages.
apropos *word*	Search for **man** pages that include *word*.
whatis *topic*	Displays a short description of topic, if a **man** page exists for *topic*.

Moving around a man page

SPACE BAR	Move forward one screen
RETURN	Move forward one line
b	Move back one screen
q	Quit

Getting help from the man pages

What are the UNIX man pages?

The UNIX **man** (manual) pages are a complete on-line reference to the system commands. To use the **man** pages, you specify the command you are looking for, then scroll through the documentation as it is displayed.

This section shows you how to select a man page and how to read the **man** pages.

Request a man page

Suppose you want to see how to use the **ls** command. Instead of reading this book, you could read a summary of **ls** right on your computer. Here's how to get the **man** page for **ls**:

```
$ man ls
```

In fact, this is how you get a **man** page for any command. You type **man** followed by the command name. If you don't know the name of the command, see *Using apropos to search for man page topics* later in this chapter.

Move around in a man page

Moving around in a **man** page is just as easy as requesting one. All **man** pages are displayed using the **more** command. **more** lets you step through large files, one screen at a time. Here's a brief summary of the commands you need to move around in the **man** pages:

Option	What does it do?
SPACE BAR	move forward one screen
RETURN	move forward one line
b	move back one screen
q	quit, return to shell prompt
?	get online help for **more**

You can also specify the exact number of lines you want to move forward by typing a number before pressing RETURN.

Reading a man page

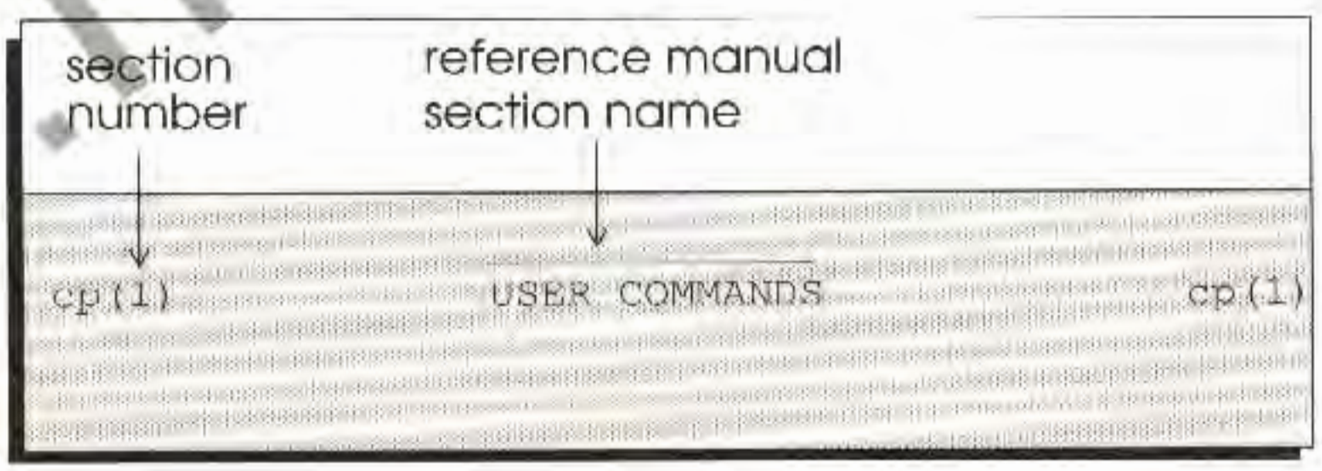
Every **man** page uses the same basic layout:

Heading	Description
NAME	command name and summary
SYNOPSIS	how to use the command (1 to 3 lines)
DESCRIPTION	explains options and their effects
examples	shows how to use some of the options

You can also type **man man** to get instructions for using the **man** pages.

Categorizing the man pages

The **man** pages are grouped by the type of information they contain: user commands, system maintenance commands, etc. For every group of **man** page commands, there is a corresponding AT&T reference manual of the same name. The AT&T reference manuals have the same information as the man pages. The title on each man page tells you which section (group) the command belongs to and the section number.



Different categories of man page information

The following table summarizes the man page categories.

Category	Type of information
1	user and system administration commands
2	system calls
3	library calls
4	file formats
5	miscellaneous
6	games
7	special files
8	system maintenance procedures

Where are the man pages stored?

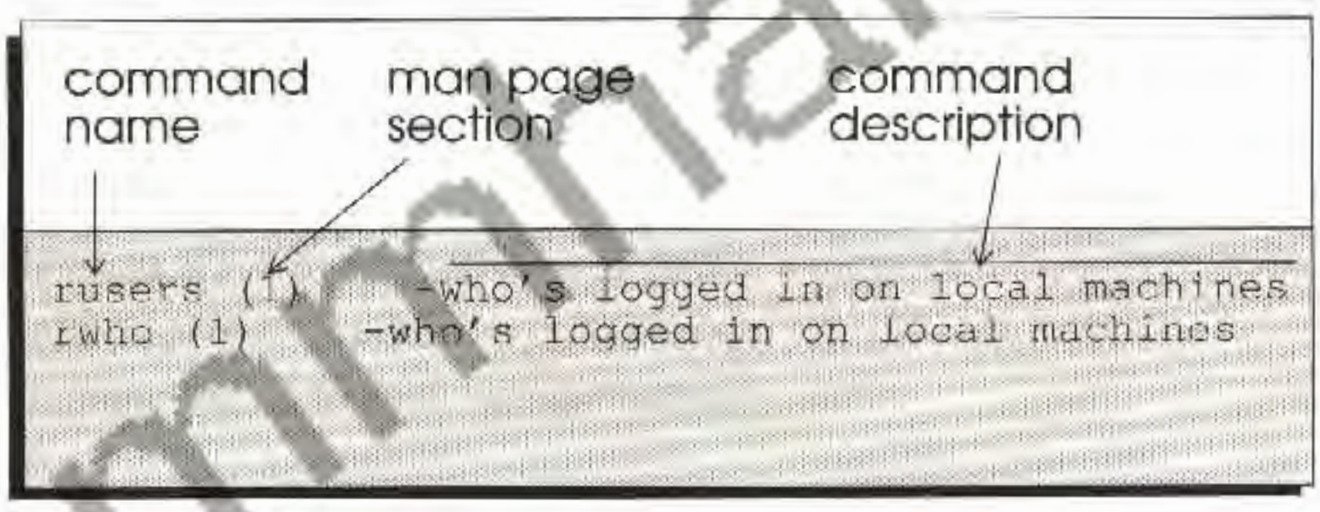
The **man** pages are stored in a number of directories under /usr/man (g1 through g8). If you want to see a list of **man** page commands for any section described above, check its directory.

Using apropos to search for man page topics

There may be times when you want to find all **man** pages that reference a particular command or word. You can use a command called **apropos** to search the **man** page directories for a keyword.

```
$  apropos who
rusers (1)  -who's logged in on local machines
rwho (1)    -who's logged in on local machines
rwhod, in.rwhod (1M) - system status server
w (1)   -who is logged in, and what are they doing
who (1)  -who is on the system
whoami (1) -display the effective current user name
whodo (1M) -who is doing what
whois (1)    Internet user name directory service
```

In the above example, **apropos** found every man page that makes reference to the **who** command or to the word who. The command name and section number that **apropos** displays come from the man page description.



Using whatis to get a command description

The **whatis** command displays three pieces of information about any command you specify:

- command name
- command section
- 1 line description

```
$  whatis who
who (1)       -who is on the system
```

Speeding up the man pages

The first time you use a **man** page, it takes a few seconds for **man** to format it. The second time you use the same **man** page, it appears on your screen almost instantly. This is because **man** saves the formatted version and uses it in the future.

Using catman to preformat the man pages

To skip the reformatting process for every new **man** page, you can format them all in advance using the **catman** command. The drawbacks to **catman** are that it creates about 15 megabytes of formatted man pages (in addition to the 5 Megabytes of unformatted **man** pages) and it takes about three hours to process.

Your formatted **man** pages gradually grow as you request more **man** pages. You can periodically remove these files from /usrcatman to free disk space; keep the ones you read most often.

Working with files and directories

File commands

cp	Duplicate files
mv	Move or rename files
rm	Delete files
chmod	Change file permissions
cat	Display file contents
more	Display a page at a time
tail	Display last 10 lines of file
head	Display first 10 lines of file
find	Search for a file

Directory commands

cd /	Move to root directory
cd	Move to your home
cd ..	Move up one directory level
pwd	Show your current directory
mkdir	Make a new directory
rmdir	Remove empty directory
chmod	Change directory permissions
ls	List directories contents

File and directory concepts

File types	Directories, devices, executable programs, text, data
Wildcards	Character substitutions. * for many characters, ? for one character in a file name
Permissions	You can give people the authority to read, write, or execute (rwx) your files and directories. Assign permissions to yourself (u), people in your group (g), others (o), or all (a) people. Use **chmod** to change the permissions. For example, **chmod a+rw** lets anyone read and write a file.

Working with files and directories

Why should you read this chapter?

Your understanding of files and directories is critical to working with UNIX. Just about everything you do with UNIX requires at least one file. You have to know how to manipulate these files by looking at, copying, moving, renaming, and deleting them.

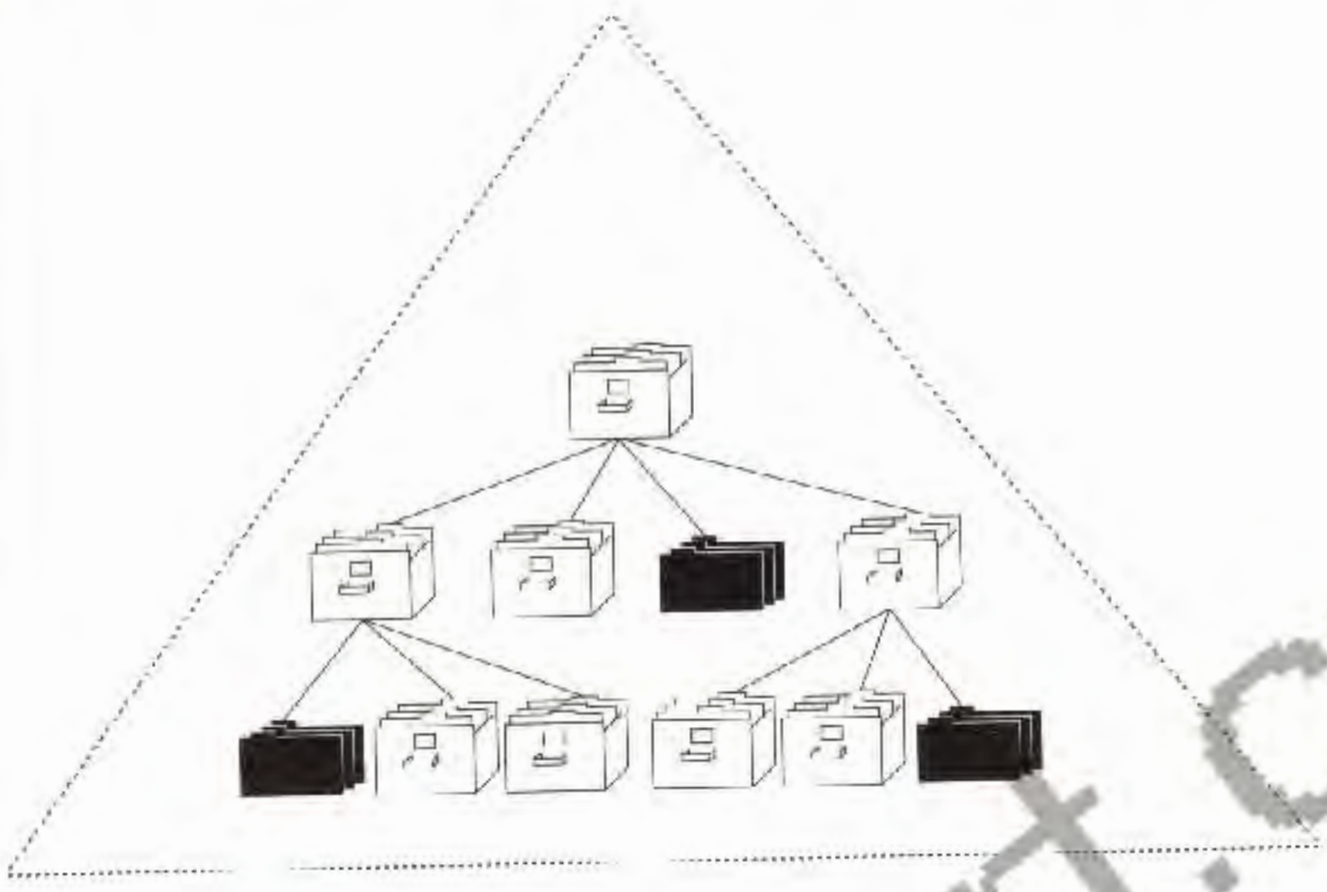
Files are all stored in directories. Some of these directories hold system files, some hold your files, and some hold other users' files. You have to be able to move around these directories, create new ones, and remove old ones to work with UNIX files.

This chapter gives you the basic information you need to work with UNIX files and directories.

Working with directories

The root directory

The UNIX directory structure resembles a pyramid.

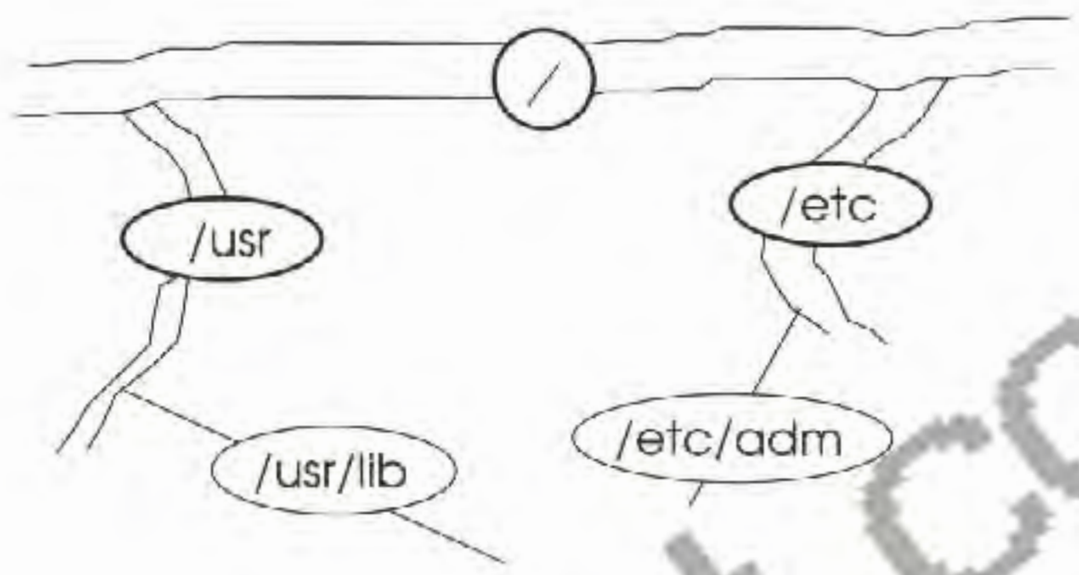


Directory structure grows downward into more directories and files

There is a "peak" or top directory called the root, and referred to in UNIX as a slash (/). Everything else branches down and out from this peak.

What is a path?

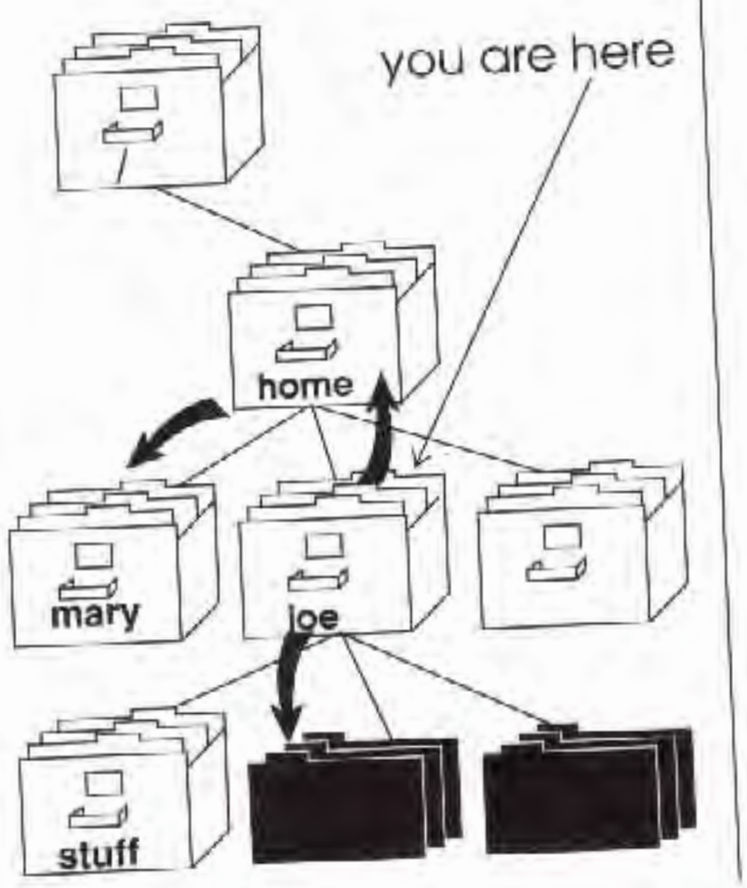
To move through the branches, you must specify a path. The path is a road map to the directory you want.



cd / moves to the root directory

No matter where you are, you can always switch to the root directory by typing **cd /** . (**cd** stands for change directory).

```
$ cd /
```



cd
move to your home

cd ..
move up one level

cd stuff
move down one level

cd /home/mary
move directly to a directory

cd ../joe
move up, then down

Different ways of using cd to move around

Your home directory

When you first log in, you are in your home directory. Your home directory is where you keep all your personal files. Your home directory also contains some special configuration files that your shell reads when you log in.

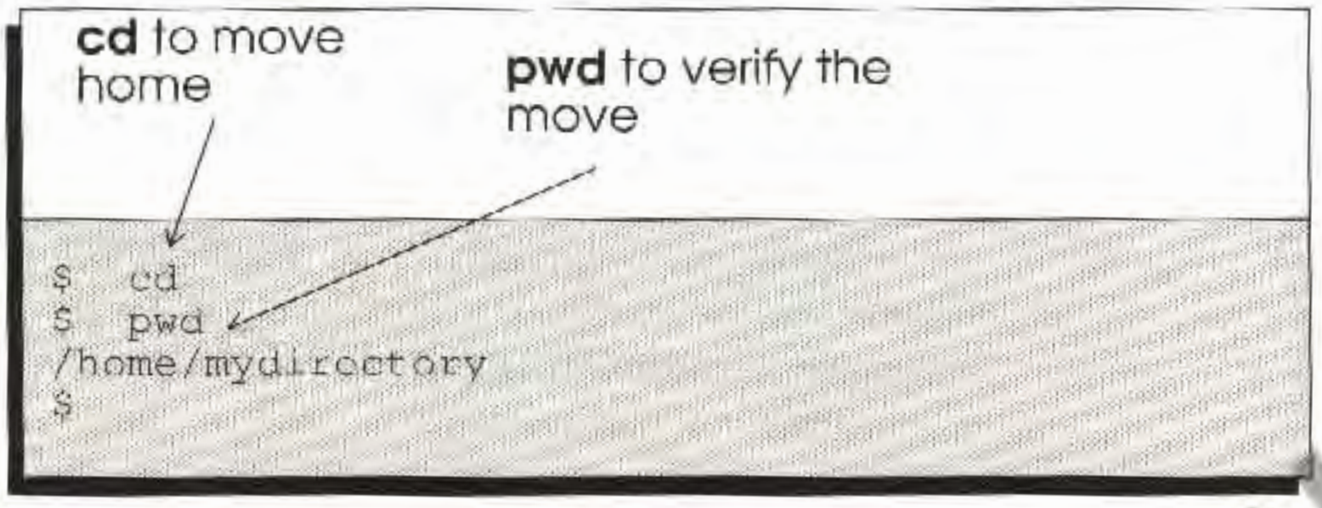
pwd to see where you are

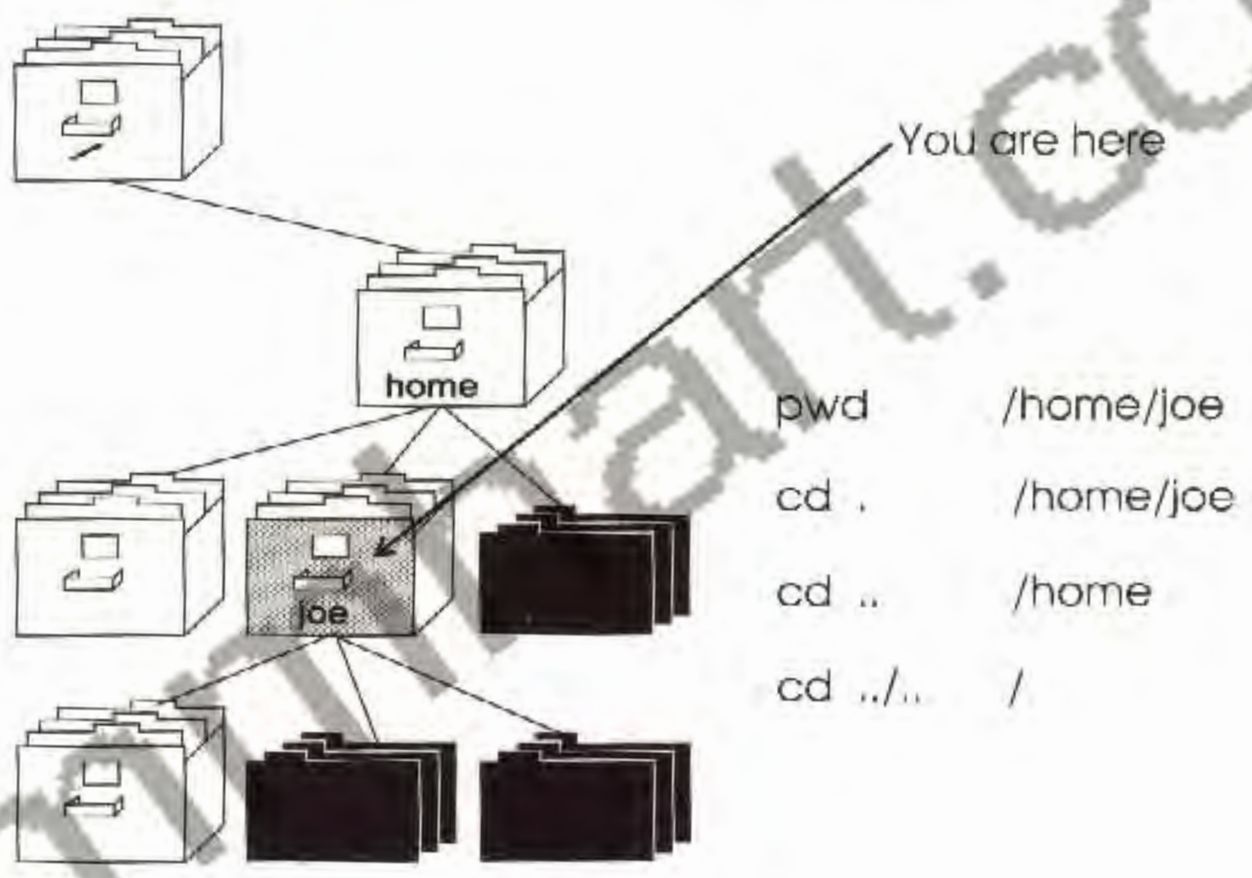
You can check that you are in your home directory by using the **pwd** (print working directory) command. **pwd** always shows your current location in the directory tree.

```
$   pwd
/home/joe
```

cd to return home

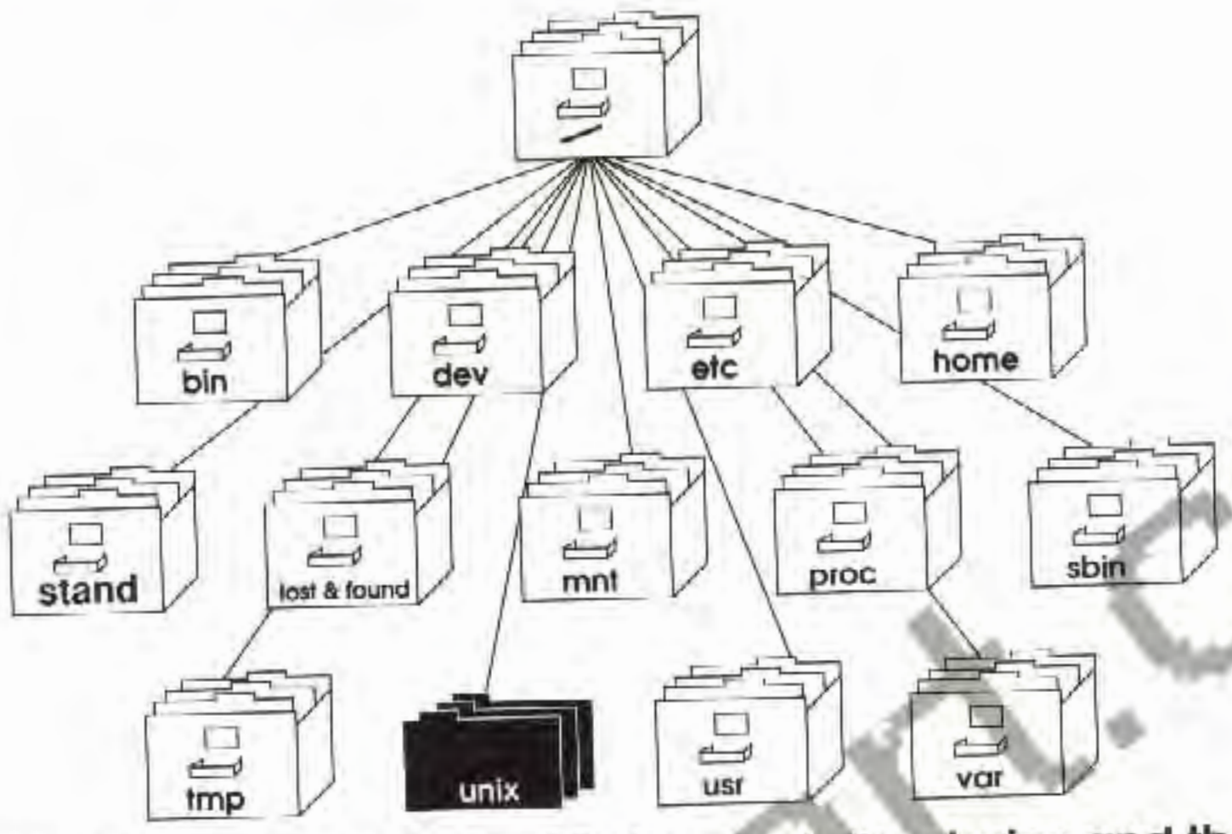
You can always return to your home directory from anywhere by typing **cd** by itself.





What are the standard directories?

System V Release 4 comes with many default directories. These start with 12 directories off root, then many others below them.



root directory contains 12 standard directories and the UNIX kernel

You can move into any one of these directories. If you want to move into the /etc directory, type **cd /etc**.

```
$ cd /etc
```

Each default directory serves a specific purpose. The table on the following page lists the directory name and its purpose.

What do the standard directories contain?

Directory	What is it for?
bin	user commands
etc	system configuration files
lost+found	file system repairs
sbin	system level commands
home	users' home directories
mnt	temporary mount point
tmp	temporary files for the system
usr	more commands and system files
dev	all the device files
proc	mount point for "proc" file system
var	system configuration files

Create directories

You can create your own directories by using the make directory command (**mkdir**). If you want to create a directory off your home directory, **cd** to your home directory, then type the **mkdir** command.

```
$ mkdir directory
```

You can change from the current directory into the new directory by using **cd**.

```
$  cd directoryname
```

Remove directories

You can remove a directory just as easily as you created one. The only catch is that the directory must be empty. If you try to remove a directory that still has files or subdirectories in it, **rmdir** tells you that the directory is not empty. You have to remove everything from a directory before you can remove the directory itself.

```
$  rmdir directory
```

Understanding the different UNIX file types

How many types of files are there?

There are three basic UNIX file types: directory, plain, and special. You can use these files to represent many different things, including:

- directories
  store pointers to other files
- devices
  files that communicate with disk drives, terminals, etc
- executables
  programs that you can run
- text or data
  contain code, characters, and data
- special
  files that are devices

Check the file type

You can use the **file** command to check a file's type.

```
$  file /etc/passwd
/etc/passwd:    ascii text
```

The **file** command won't work on files unless you have permission to read them. If you try to use **file** on a file that you don't have permission to read, the **file** command responds with an error message.

```
filename:    cannot open:Permission denied
```

If you get this message, you are trying to look at an important file with restricted permissions. You can't look at this file unless its owner lets you.

Listing files

The **ls** (list) command gives you a list of files in a directory. There are at least twenty options you can use with **ls**. We're going to describe some of the more common options here, but you can use the manual pages (type **man ls**) to get information about other **ls** options.

List your files alphabetically

ls, without any options, lists the files in the current directory in alphabetical order by file name.

```
$  cd /
$  ls
bin     etc       lost+found      sbin
home    mnt       tmp             usr
dev     stand     proc            unix
var
```

List one file

ls with a file name shows you only that file (if it exists). **ls** with a directory name shows the files in that directory.

```
$  ls filename
filename
$  ls directoryname
file    file    file    file    file    file
file    file    file    file    file    file
file    file    file    file    file    file
```

Other ways to list files

You specify options for the **ls** command by typing a space, a dash, and the option.

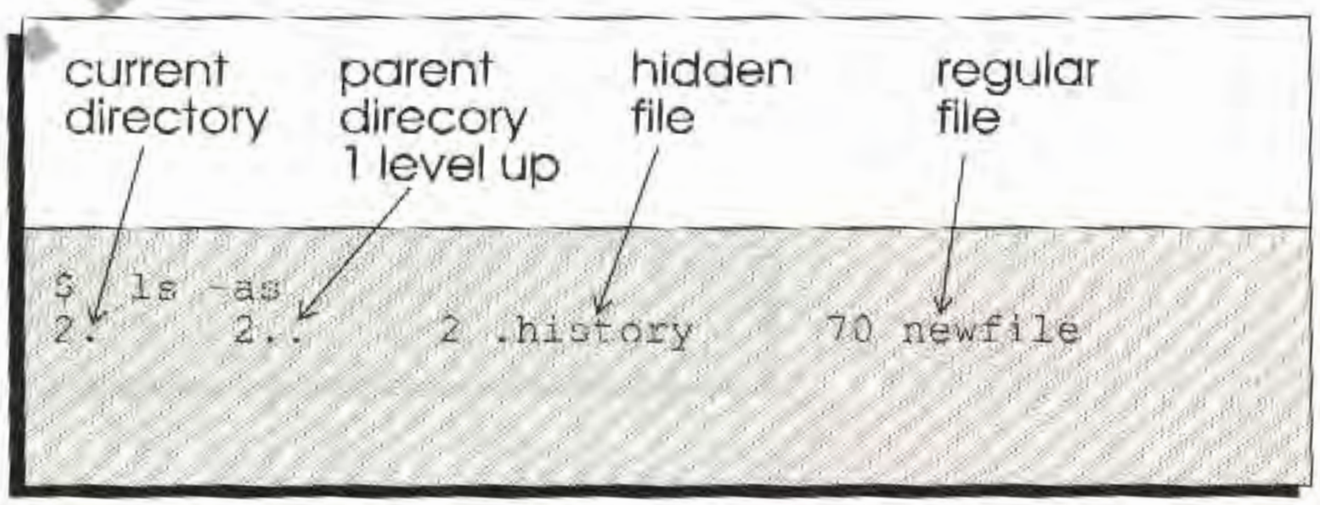
```
$  ls -l
```

Here's a list of some useful **ls** options:

Option	What does it do?
-a	Display all files in the directory
-l	Show type, size, permissions, owner
-s	Display block size of each file
-t	Display in order by last change date
-u	Show time last used
-r	Reverse the sort order

Combine ls options

You can combine some **ls** options to customize your file list even further. You could combine the **-a** (all) and the **-s** (block size) options to list all files and their size in 512 byte blocks.



The best way to learn about the **ls** options is to try the ones we gave you, then experiment with different combinations. You can't hurt anything by experimenting with **ls**. If you don't have enough files in your directory, try listing the /usr/lib or /usr/public directories, then any directories under them.

Finding files

Search for files

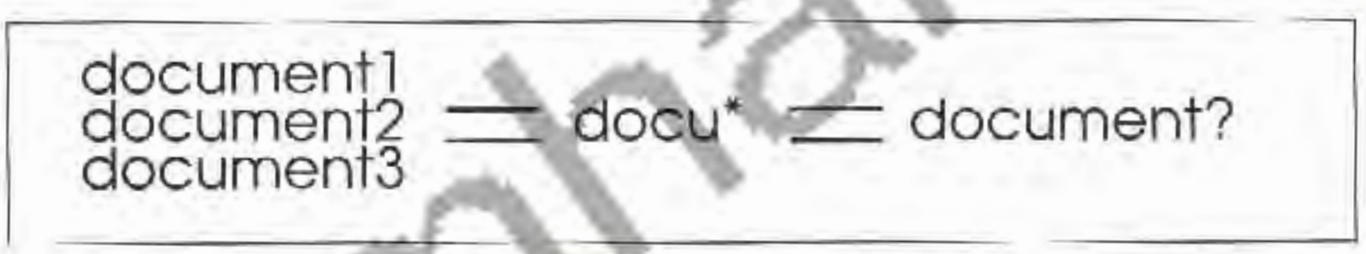
The UNIX system has commands that search through directories for files. This section shows you how to:

- substitute wildcard characters in file names
- use the **find** command to locate files

Use * and ? to match patterns

The shell lets you substitute characters in a file name with the asterisk (*) and the question mark (?), as well as other special characters. These characters are known as wildcards because they can represent any characters you want.

The difference between the two is that the asterisk can substitute for any characters in a file name, but the question mark must substitute for exactly one character at a time.



Your home directory might contain 50 files. You only want to list document.1, document.2, and document.3. Here's two ways the wildcard characters help you do this:

```
$  ls doc*.*
document.1   document.2   document.3   doc.review
```

A better way is to use a more exact search pattern.

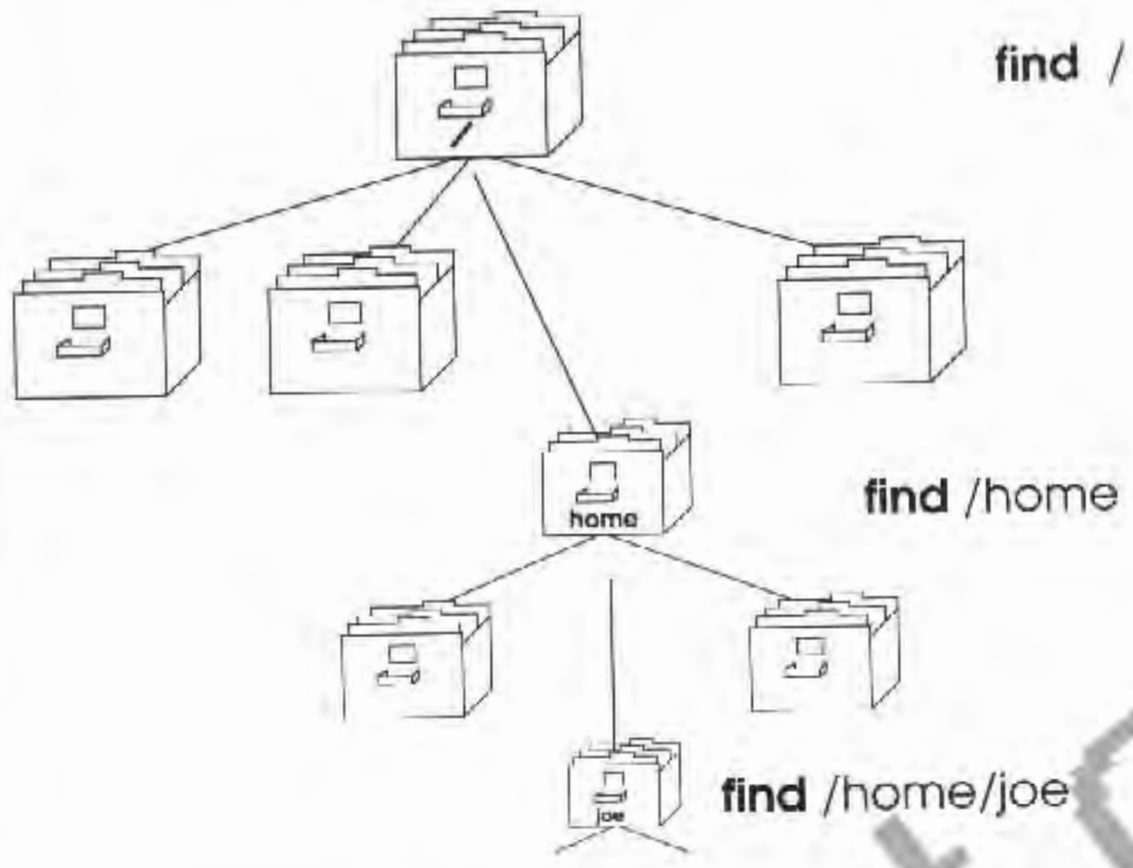
```
$  ls document.?
document.1  document.2  document.3
```

As you can see, the second method, using the question mark, is more specific than the asterisk method. You weren't looking for a document called doc.review, but because asterisks are a general substitution, you got it anyway.

Use the find command to locate files

The **find** command, like many UNIX commands, is very powerful. You can use it to find any number of files, anywhere on your hard disk, in any directories that you are allowed to read. The power and flexibility of the **find** command also make it difficult to use. For this reason, we are going to show you the basics of using **find** to locate a file. If you want to experiment further with **find**, check the man pages (type **man find**) for more specific uses.

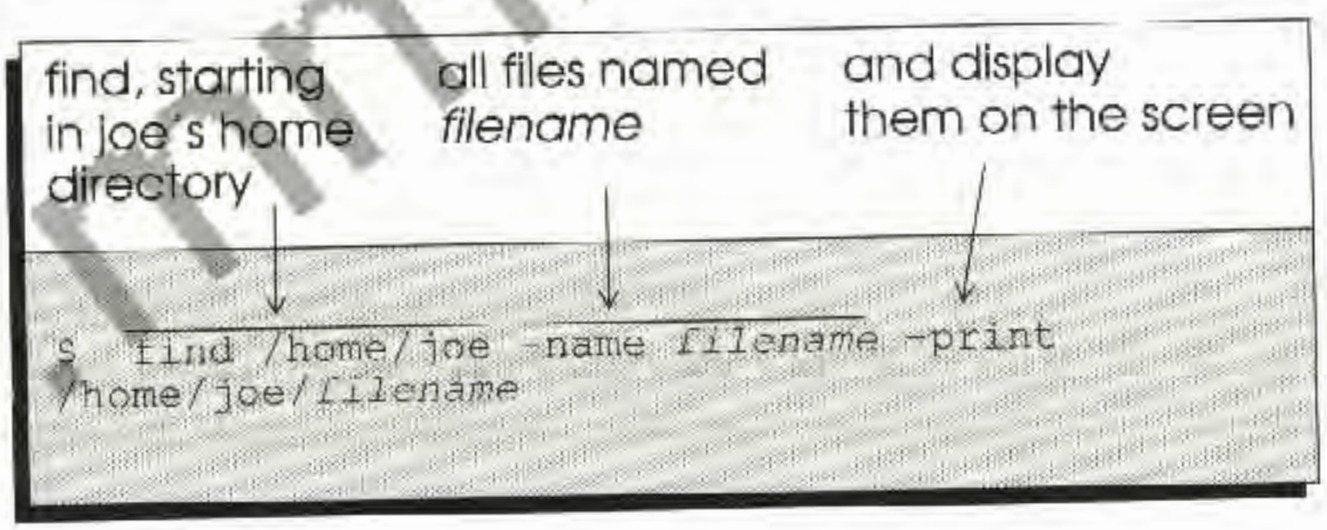
The **find** command searches through every directory and every disk in your machine if you start from root. This takes a long time and could produce an overwhelming amount of information. Most of the time you will probably want to search for files starting in your home directory, or in a specific directory branch. In this case, **find** starts from the specified directory and searches through the directories that branch off it.



find can search any part of a directory tree

Find files in or below your home

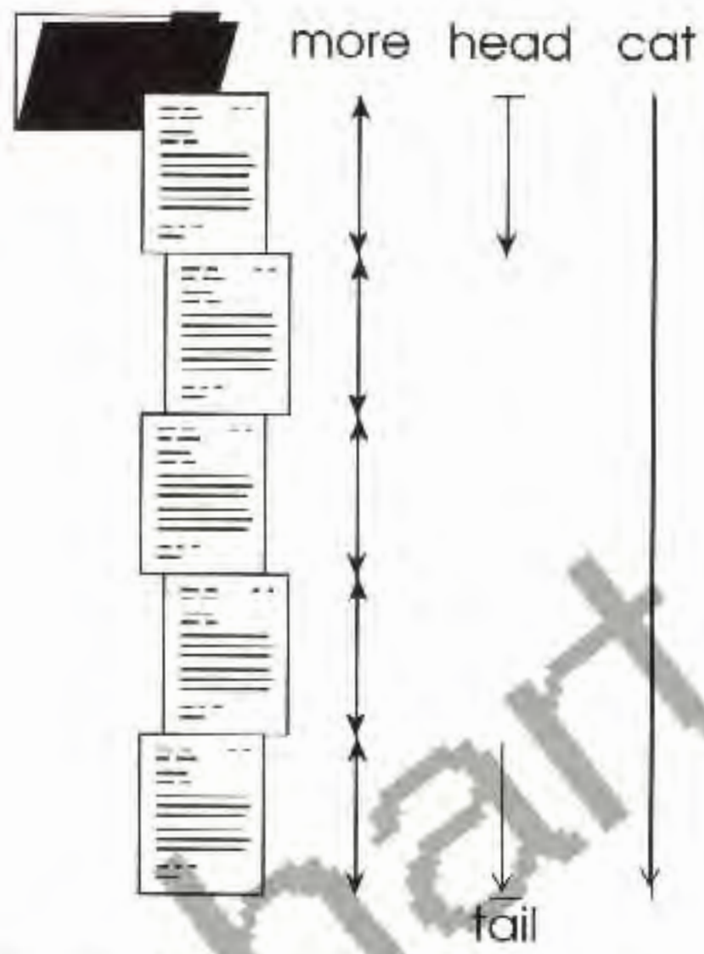
Here's the most basic use for the **find** command:



Looking at a file's contents

Display files

You can look inside a text file without actually doing any work on it. This section gives you some options for looking at all or part of a file (top, bottom, or page by page).



Different commands look at all or part of a file

Display an entire file - cat

Use the **cat** command to display the contents of one or more files.

```
$  cat filename
```

If you type **cat *filename***, and the file does not exist, **cat** tells you that it can't open the file. If you try to **cat** a file that you are not allowed to read, **cat** tells that you are denied permission.

If you are not in the directory where the file is located, you must type the path to the file before the filename.

```
$  cat /users/home/joe/filename
```

Display a page at a time - more

Large files scroll by quickly, leaving only the last screen of lines visible. Unless your file is very small, you miss the top and middle parts of it. The easiest way around this problem is to use one of the other commands described in this section. The hard way is to use CTRL-S to stop scrolling and CTRL-Q to restart it.

The **more** command works like **cat**, except that it pauses after each screen and lets you move forward or backward. You press the space bar to continue on to the next screen. **more** shows the percent of the document you have read in the bottom left corner of the screen.

more shows one page at a time

```
$  more filename

top of screen

bottom of screen
-more- 50%
```

more options

more has many other options for moving around the file. Some of these options are listed below.

Options	What does it do?
SPACE BAR	Move forward one screen
RETURN	Move forward one line
b	Move back one screen
q	Quit, return to shell prompt.
?	Get online help
/text	find text pattern

You can also specify the exact number of lines you want to move forward or back. For other options, check the **more** man pages.

Display the end of a file - tail

The **tail** command displays the tail, or end, of a file. The **tail** command, in its simplest form, takes just a filename.

```
$  tail filename
```

By default, **tail** displays the last 10 lines of filename. You can also specify the number of lines you want to see by typing + or - number of lines.

```
$  tail +3 filename
```

If you type a negative number, -3, tail shows the last three lines of the file. If you type a positive number, +3, tail starts the display at line 3.

See the **tail** man pages for a complete list of **tail** options.

```
$  more filename

top of screen

bottom of screen
-more- 50%
```

more options

more has many other options for moving around the file. Some of these options are listed below.

Options	What does it do?
SPACE BAR	Move forward one screen
RETURN	Move forward one line
b	Move back one screen
q	Quit, return to shell prompt.
?	Get online help
/text	find text pattern

You can also specify the exact number of lines you want to move forward or back. For other options, check the **more** man pages.

Display the end of a file - tail

The **tail** command displays the tail, or end, of a file. The **tail** command, in its simplest form, takes just a filename.

```
$  tail filename
```

By default, **tail** displays the last 10 lines of filename. You can also specify the number of lines you want to see by typing + or - number of lines.

```
$  tail +3 filename
```

If you type a negative number, -3, tail shows the last three lines of the file. If you type a positive number, +3, tail starts the display at line 3.

See the **tail** man pages for a complete list of **tail** options.

Display the beginning of a file - head

The **head** command displays the head, or beginning, of a file. The **head** command, in its simplest form, takes just a filename.

```
$  head filename
```

head displays the first 10 lines of filename. You can also specify the number of lines you want to see by typing minus and a number.

```
$  head -20 filename
```

This example displays the first 20 lines of *filename*. You can specify any number of lines for **head** to display.

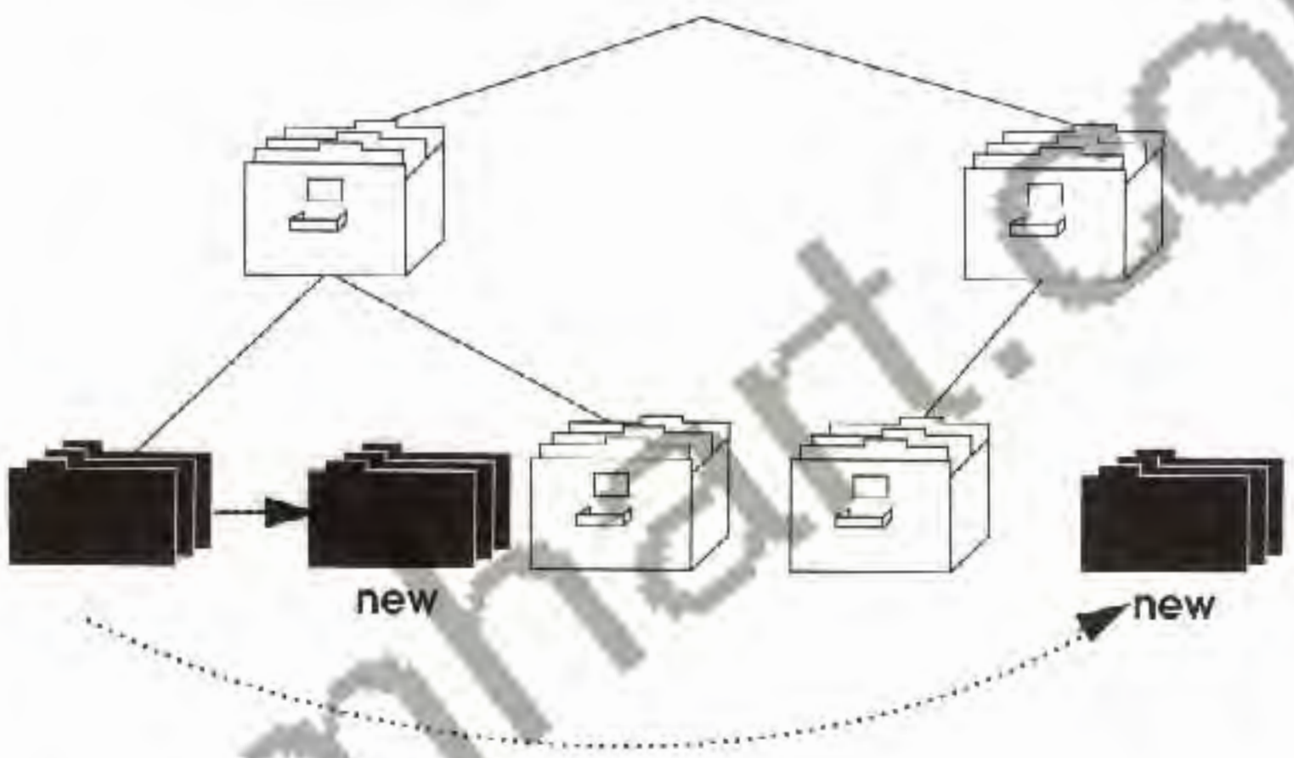
See the **head** man pages for a complete list of **head** options.

Copying files

Copy files

The **cp** (copy) command lets you duplicate a file and give it a new name. Although you use **cp** exclusively for duplicating a file, it can actually achieve three different goals:

- duplicate a file and give it a new name
- duplicate a file and put it in a different directory
- duplicate a file, put it in a different directory, and give it a new name.



cp copies files to a new name or a new place

You can also **cp** more than 1 file, by typing each filename or by using wildcards.

Same directory-different name

The following example shows you how to duplicate a file.

```
$  cp sourcefile destinationfile
```

The source file is the one that already exists, and the destination file is the duplicate you are creating. This example would put the new file in your current directory.

NOTE: If the destination file already exists, **cp** overwrites it, destroying its original contents. Don't be careless; if you accidentally copy the wrong direction you could destroy a file instead of copying it.

Different directory - same name

Copying files to a new place is similar to the above example, except you must specify a destination directory.

```
$  cp sourcefile directory
```

Different directory - different name

This example shows you how to copy a file between directories, giving the new file a different name.

```
$  cp sourcefile directory/newfile
```

Copy several files at once

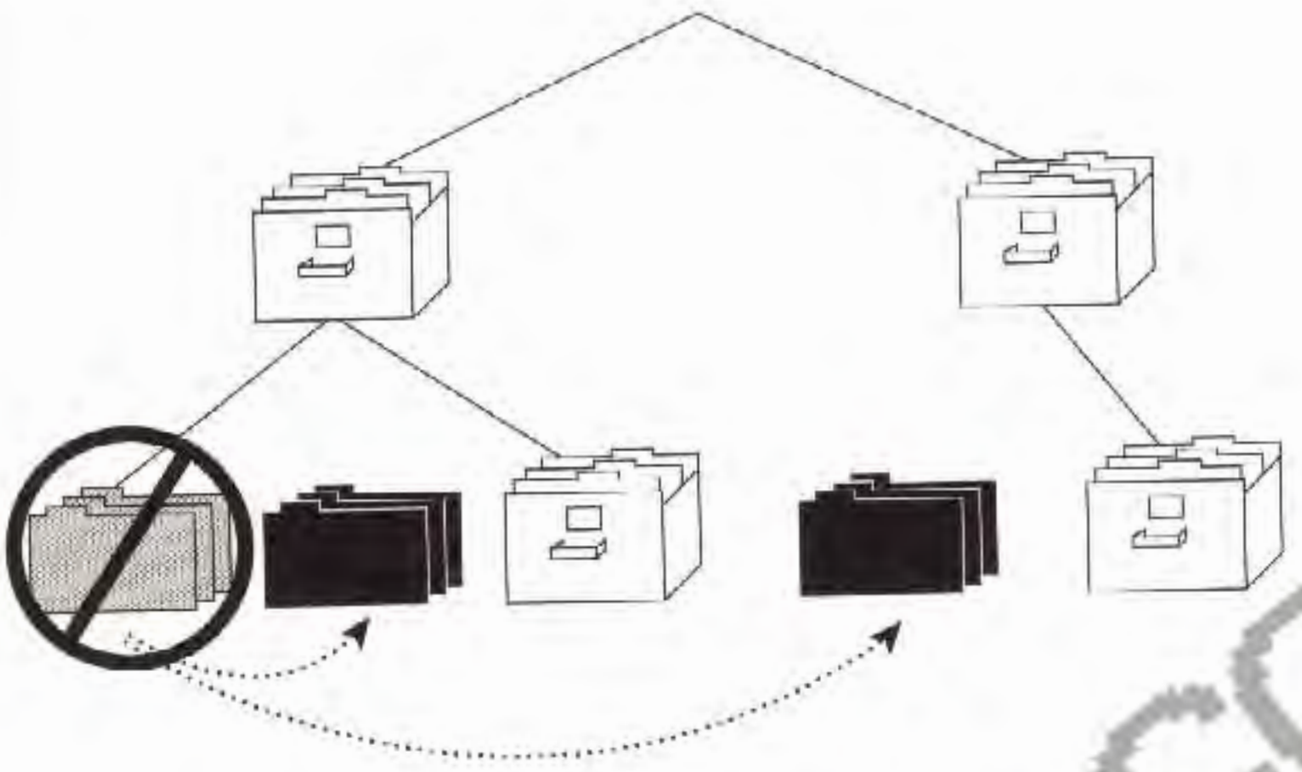
This example shows you how to copy several files between directories.

```
$  cp file1 file2 file3 directory
```

You use the move (**mv**) command to rename files or move them to a different place.

Renaming and moving files

Rename files



mv renames a file or moves it to a different place

Unlike the **cp** command, **mv** does not create a duplicate of the first file. It replaces the first file with the new file, so you have a new file but not the old one.

```
$  mv filename newfilename
```

NOTE: If the new file already exists, **mv** overwrites it, destroying its original contents.

Move files to a different directory

You can use **mv** to move a file to a different directory. **mv** does not create a duplicate file for the destination directory. It takes the original from the source directory and leaves nothing in its place.

```
$ mv filename directory/filename
```

Move many files at once

You can move multiple files to a new directory by listing more than one file after the **mv** command and before the directory name.

```
$ mv filename filename2 filename3 directory
```

Deleting files

The rm command

You can delete files using the remove (**rm**) command.

```
$  rm filename
```

Don't let the simplicity of this command fool you. It has the potential to do a lot of harm. You should double check the filename before using **rm**. You should be even more cautious when you delete files using wildcard characters. You could end up deleting more files than you wanted to.

rm does not ask you to confirm that you want to delete the files, nor does it tell you which files it deleted. You might want to use **rm -i** instead; it asks if you are sure before deleting each file.

NOTE: If you remove an important file, there is no way to get it back. Don't be careless with the **rm** command.

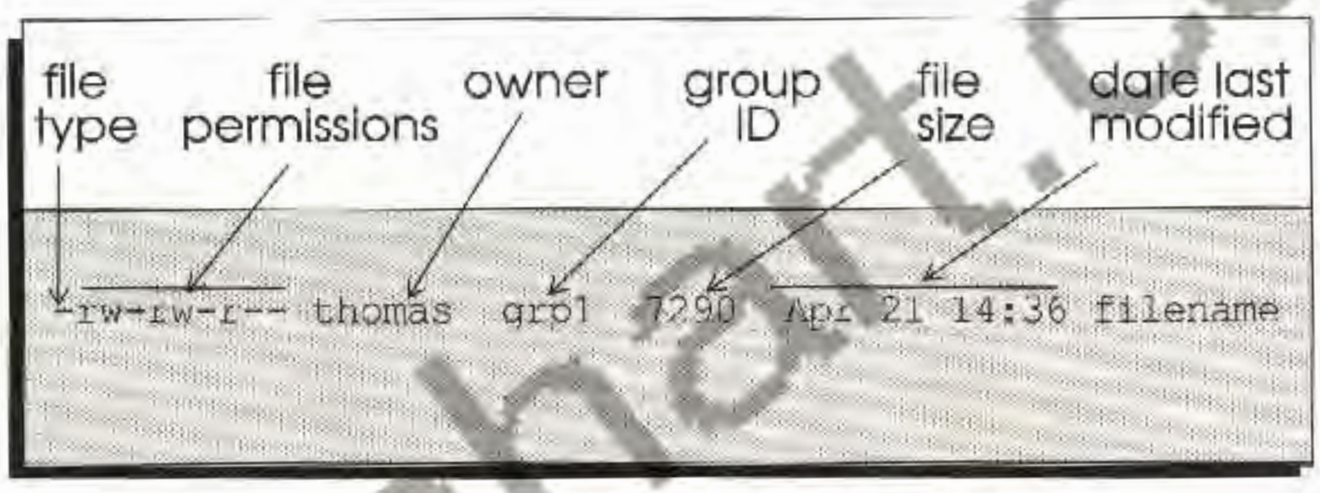
Protecting your files from other users

File attributes

If you check your directory using **ls -l** you see lines of information that include several different groups of data.

```
$  ls -l
-r w-rw-r--  1 thomas grp1 7290 Apr 21 14:36 filename
```

These are the file attributes. The following figure shows you what each attribute means.



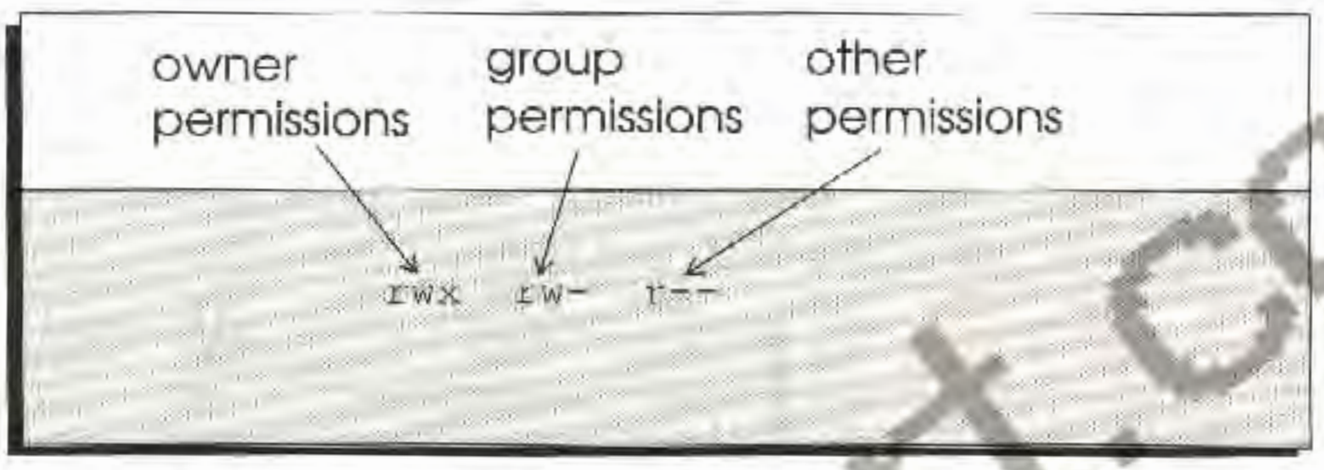
ls -l shows everything about a file

Attribute	What does it mean?
file type	Regular(-),dir(d), block(b),char(c), pipe(p), or link (l)
permissions	Who can use the file and how
owner	Person who created the file
group ID	Group to which owner belongs
file size	Space it takes up on your disk
last modified	Date and time file was last changed

Change file permissions

You can prevent other users from reading, writing, or executing your files. You do this by changing the permissions associated with a file or directory.

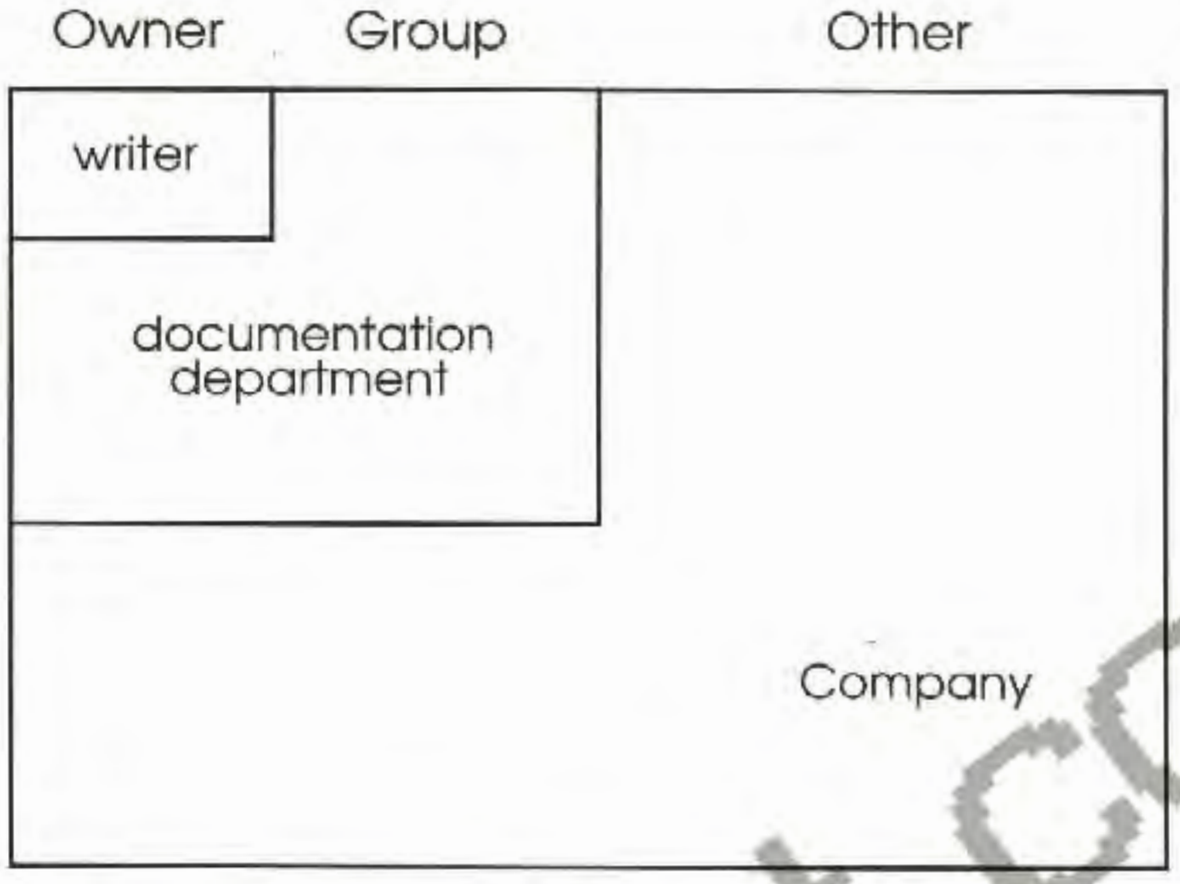
The letters from position 2 to 10 at each line indicate the file permissions. Notice that there are three groups of three characters (rwx rwx rwx). Each group of three letter refers to specific types of users.



Permissions can be different for you, your group, and all others

The above example says that the owner has read, write, and execute permission for this file. People in the owner's group can read and execute it, and everyone else can only read it. There are other less common letters you can substitute here, but they are generally used by programmers.

Symbol	Gives permission to
r	read the file
w	write the file (change it)
x	execute, or run, the file
-	not available



Your world is you, your group, and everybody else

You can change the permissions for any file you own. You can make it so that:

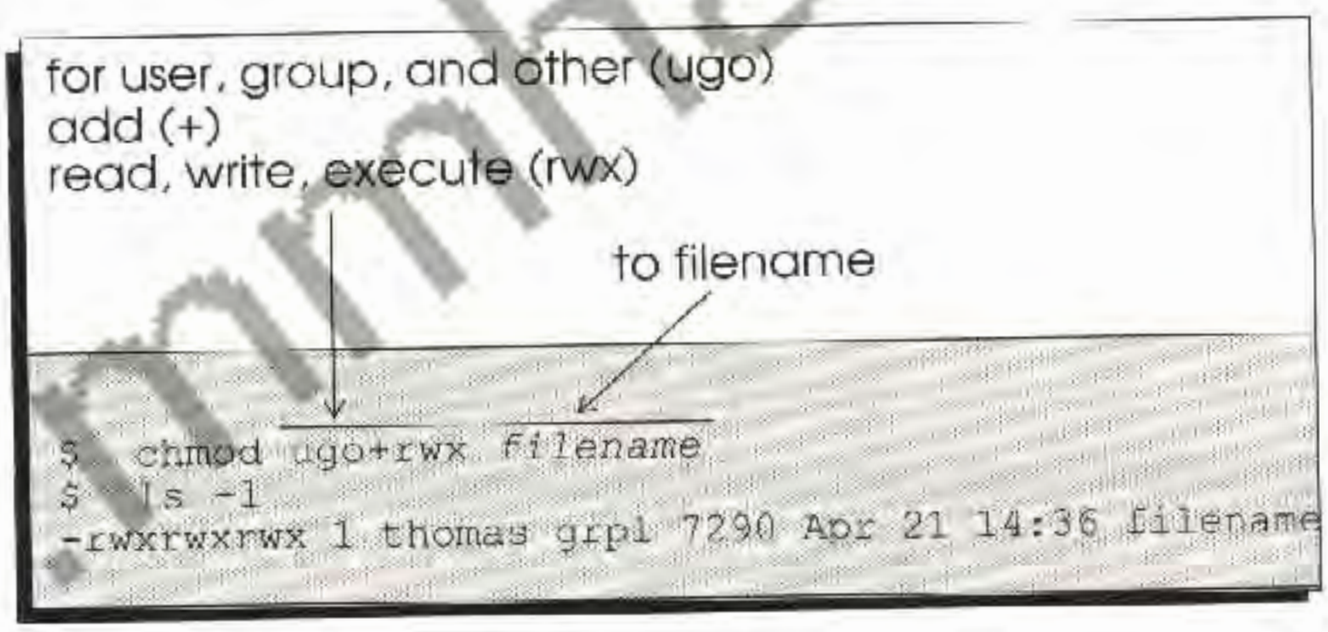
- you are the only person who can change a file, but everyone can read it (-rw-r--r--)
- only you and people in your group can change or use a file (-rw-rw----)
- nobody but you can even look at your file (-r--------)
- any other combination you can think of.

Change permissions using chmod

You change file permissions by adding or removing an r, w, or x. You add or remove an r, w, and x using the **chmod** command.

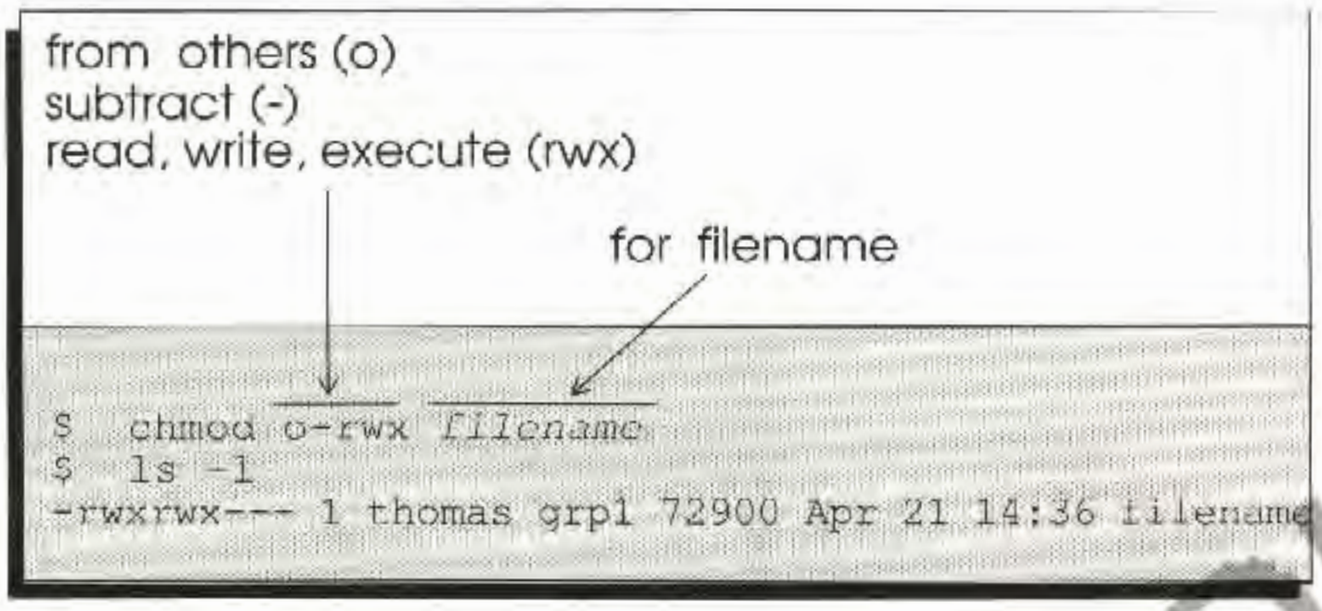
Basic use for chmod

The following example shows the basic use for **chmod**.



ls -l then shows the changed permissions.

The following is another example of **chmod**.

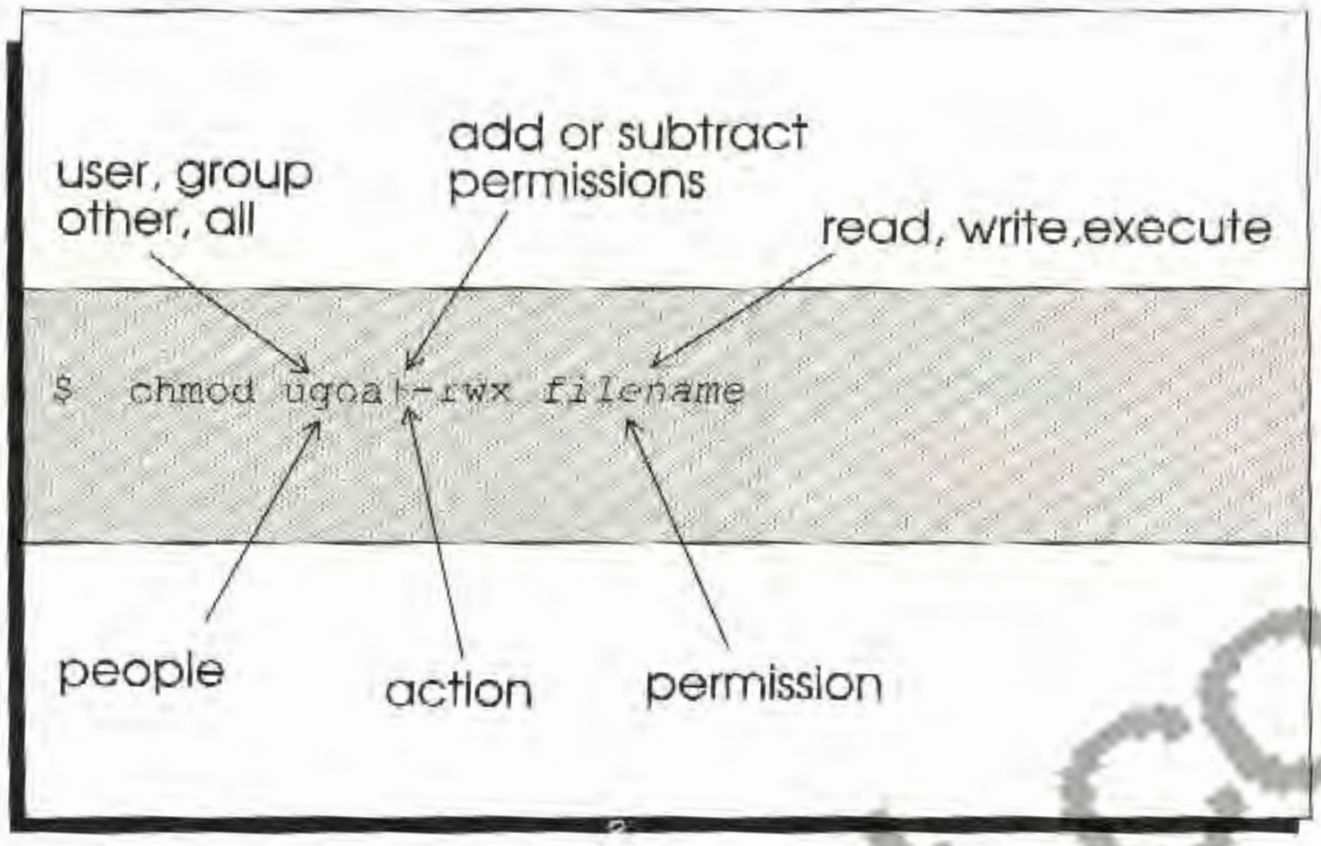


ls -l then shows the changed permissions.

As you can see from the two examples, **chmod** adds or subtracts privileges from a file. All you have to do is specify any combination of options in the form of *people action permission*.

You do not put a space between the 3 components of the permissions argument.

Using chmod on directories



You can use **chmod** to restrict your directories as well. You can make it so other users can't look in, add files to, remove files from, or use files in you directories. **chmod** for a directory works exactly like **chmod** for files, except you specify a directory name. The directory permissions only affect operations on the directory, not the files in the directory.

Symbol	Gives group permission to
r	list directory (read names in it)
w	add files to it or delete them (modify)
x	use files in the directory or execute them (search the directory)

Troubleshooting

Problem	Solution
Can't change into a directory	Be sure you are typing the directory path correctly. You may not have permission to read or execute that directory.
Can't delete a directory	Directory still has files or subdirectories in it. You may not have permission to write or execute the parent directory of the directory you want to delete.
Can't delete a file	You may not have permission to write or execute the directory where the file is.
Can't list a directory	You may not have read access to the directory.
Can't display, move, rename, or copy a file	You may not have read access to the file. You may not have write permission for the file. You may not have permission to read, write or execute the directory you are now in. You may not have used the correct path or the file might be empty.

Problem	Solution
Accidentally copy over or delete an existing file	No way to get it back. Be extremely careful when copying, moving and deleting files!
Can't find a file using find	The file may not be in the path you specified. The file may be in a directory that you can't read.
Find lists too many files	Modify your search path or wildcard pattern to narrow the search.
Which file was used last?	Use **ls -lt** or **ls -alt** to list the contents of a directory in order by date changed. Use **ls -altu** to list files in the order in which they were last accessed for any reason.
Copy an entire directory tree	Use the **cp** command with the **-r** option.

Printing

Printing terms and concepts

Print service	Amiga UNIX uses the standard UNIX **lp** print service commands
lpsched	lp program that spools print jobs and schedules them for printing
Conversion interface	Program through which files pass on their way to the printer. Used to convert text to Postscript, for example, or to send special characters to a printer
Default printer	Printer to which you automatically print
Print job	Any file that is queued and waiting to print

Print commands

lp	Command that sends a file to the default printer
lpadmin	Command you use to define a printer
lpstat	Command you use to check on the status of a print job or printer
disable	Stop printing
enable	Restart the printer
cancel	Kill a print job
reject	Do not let jobs into print queue
accept	Allow jobs into the print queue

Conversion interfaces

Conversion interfaces are located in /usr/spool/lp/model

Available interfaces:

postscript
standard

Printing

Why should you read this chapter?

Most of the effort involved in printing is setting up the **lp** (line printer) print service. This chapter shows you how to set up the **lp** print service and how to print files.

Using a printer

You (and other people who use your computer) can:

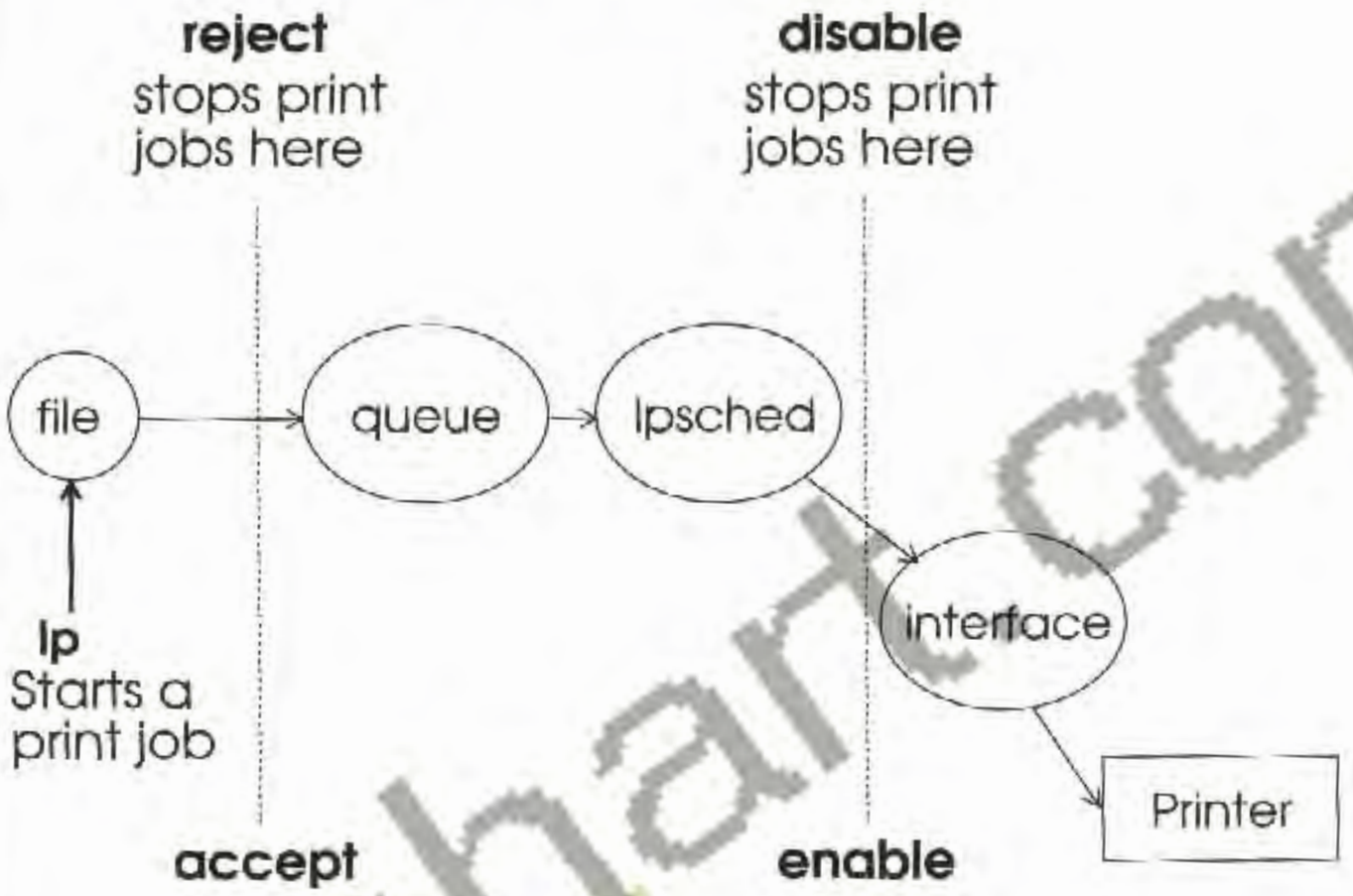
- print a file
- list the jobs waiting to print
- cancel a print job

Set up a printer

The important printer configuration steps you need to take are:

- make sure **lpsched** is running in the background
- physically connect the printer to a serial or parallel port
- login as root
- check /etc/inittab to be sure the port is not being used for logins
- name your printer
- identify the printer's device name
- choose a conversion interface depending on the type of printer you have

You use the **lp** command to send a file to a print queue (and ultimately a printer). **lp** takes your file and puts it in the queue for the print scheduler (**lpsched**). One by one, **lpsched** sends the files to the printer, until there are no more left in the queue.



Understanding the UNIX print process

Using lpadmin to add a printer

Define your printer

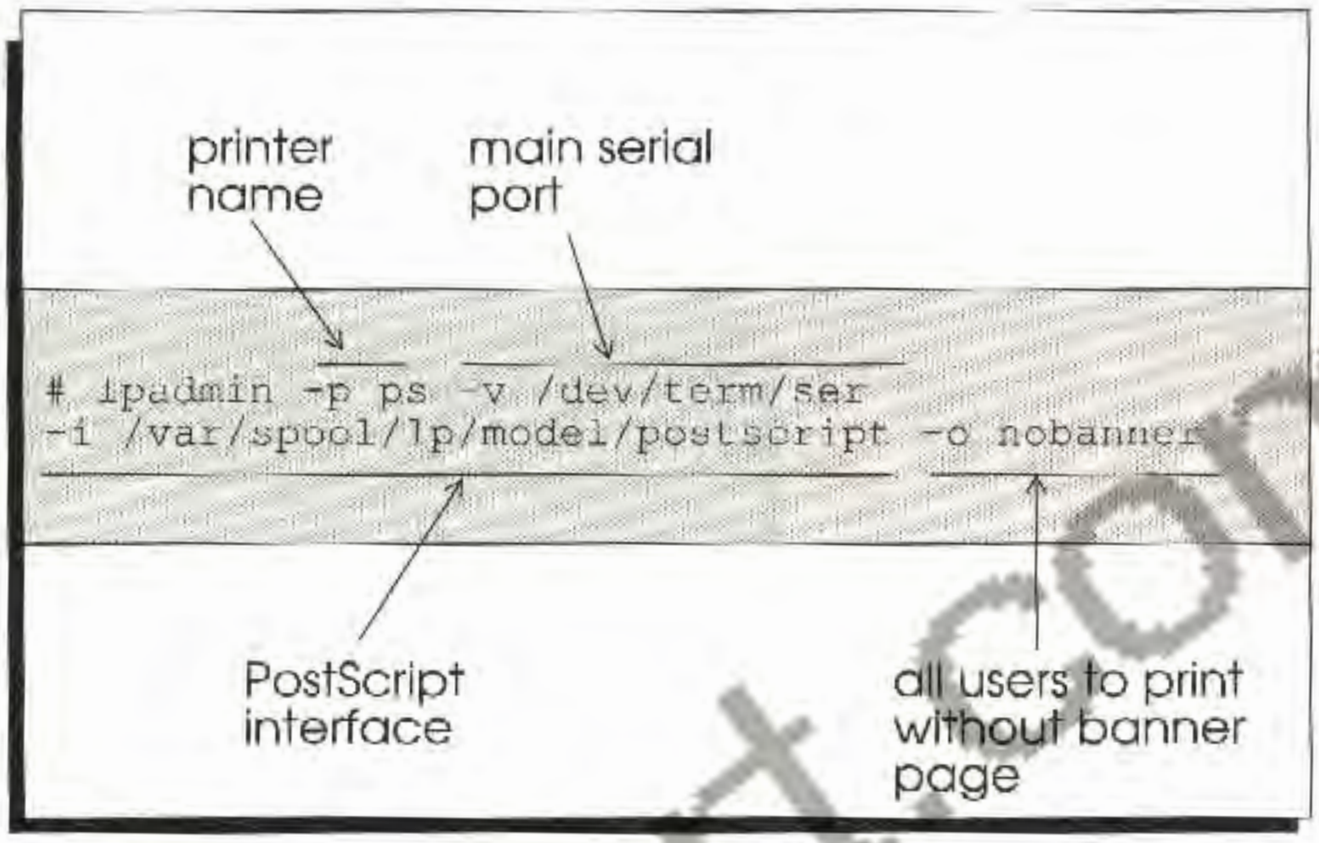
Before you can run **lp**, you need to define your printer. You do this by using the **lpadmin** command.

You use **lpadmin** to add a printer connected to your system, change an existing one, or change the default printer. The following example shows the basic format for **lpadmin**.

```
#lpadmin -p name -v device -i interface -o nobanner
```

name	what users call this printer
device	device name (/dev/par, /dev/term/ser, or /dev/term/ql00 through /dev/term/ql06 if you are using a ql card. The bottom port on the ql card is /dev/term/ql00 and the top port on the ql card is /dev/term/ql06)
interface	program that controls files for printer (standard, PostScript, or any program in /var/spool/lp/model)
nobanner	allows you to print without a banner page if you choose

The following example defines a PostScript printer on the main serial port.



enable and accept the new printer

To complete the process of adding a printer, use the **enable** and **accept** commands. These tell **lp** and **lpsched** to accept print requests from users, and to print these requests.

```
# enable printername
# accept printername
```

The default printer is the one where you want your files to print when you don't specify a printer.

In the previous example of **lpadmin**, you defined a printer and an interface but did not tell **lp** to assume that print jobs automatically go to this printer. **lp** won't know what printer to use unless you specify one with each **lp** command or assign a default printer that you use most often.

Use lpadmin to name your default printer

To define your default printer, you need to run the **lpadmin** command again with the **-d** option.

```
#  lpadmin -d printername
```

Send a print job to a different printer

The printer *printername* will be your destination printer until you change it or specify another one. You can change it at any time by running **lpadmin -d** with another printer name. You can change your default printer for a file by typing the following option when you print the file.

```
$  lp -d printername filename
```

Special note for serial printers

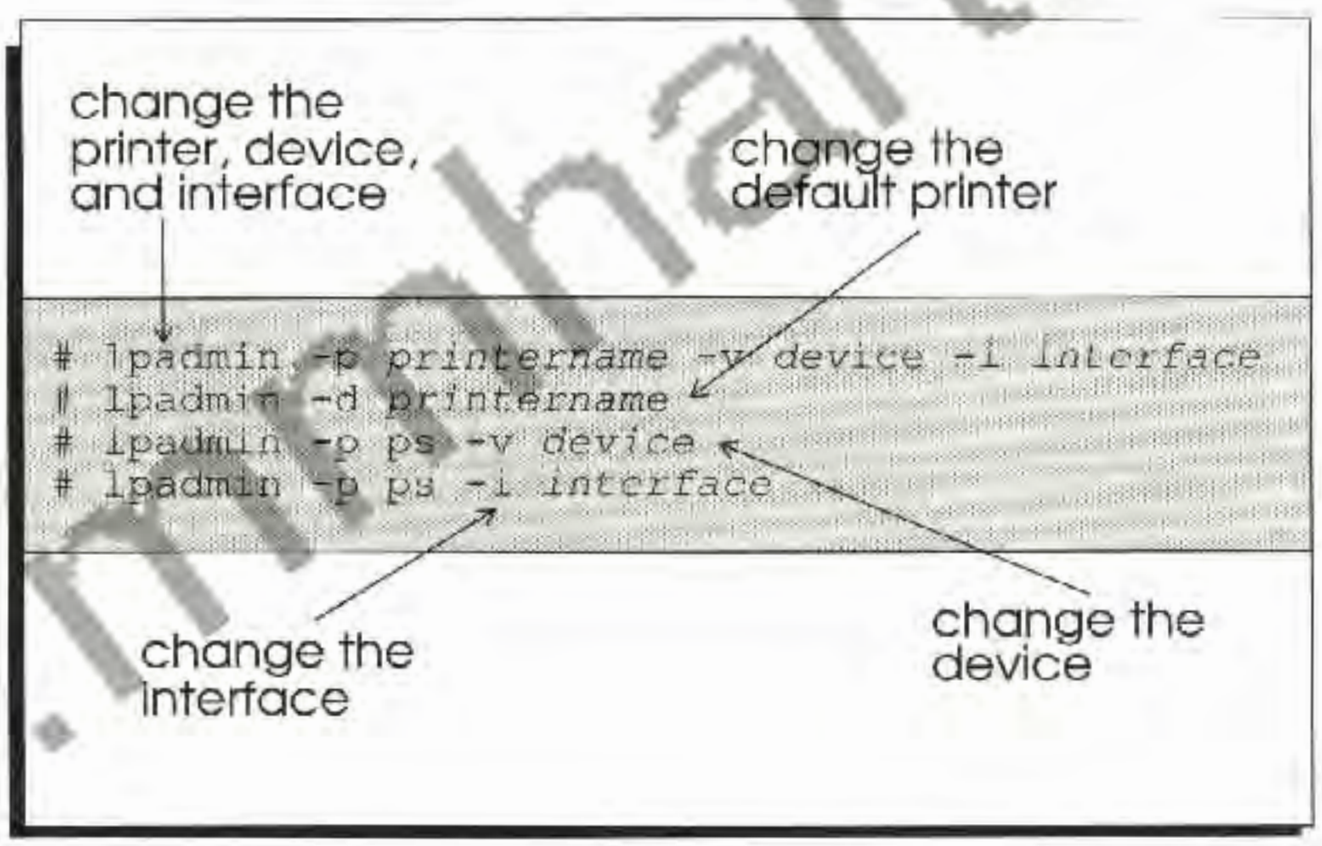
If you connect the printer to a serial port, you should check /etc/inittab. If you have a ser line in /etc/inittab, be sure it is off so you can print to it. The ser port

inittab line usually has its getty turned off; it is switched to respawn if the port is used for a dumb terminal.

```
ser0:23:off:/etc/getty term/ser # main port
```

Change an existing printer

Use **lpadmin** to change the settings for existing printers. You can change the device name and the interface, or either one, by identifying the printer and typing the new setting.



Delete a printer

To delete a printer from the **lp** system, use the **lpadmin -x** option.

```
# lpadmin -x printername
```

Printing a file

Use the lp command to print a file

The **lp** command starts the printing process. After you configure your printer with lpadmin, you can type **lp** ***filename*** to print a file.

```
$ lp filename
```

Print to another printer

If you want to use a printer other than your default, you need to tell **lp** which printer to use.

```
$  lp -d printername filename
```

You can check the default printer name by typing **lpstat -p**.

```
$  lpstat -p
printer name is idle. enabled since thur sep 6
12:25:37 EDT 1990. available.
```

lp options

Sometimes you have to specify options to the **lp** command. Suppose your default printer is a PostScript printer with a PostScript interface. This interface normally converts text files to PostScript, but you might want to print **troff** or already-formatted PostScript files on this printer. The following examples illustrate how different **lp** options print different types of files.

Print a PostScript file

```
$  lp -o postscript filename
```

The -o postscript option tells the PostScript interface not to reformat the file, because it's already in PostScript.

Print a troff file on a PostScript printer

```
$  troff filename | tpscript | lp -o postscript
```

This command uses **troff** to format a text file, then converts it to PostScript and sends it to the printer as a postscript file.

Print a text file

```
$  lp filename
```

This command prints the file using the interface for the default printer. If a text file is sent to your PostScript printer, it is converted to PostScript. If a **troff** or PostScript file is sent without options, it will not print properly, because the interface will try to convert it again.

Print without the banner page

The banner page is the first page to come out of the printer. It contains the following information about your job:

- who requested the job and from where
- id
- printer name
- date and time

Use **lp -o nobanner** to print your file without the banner page.

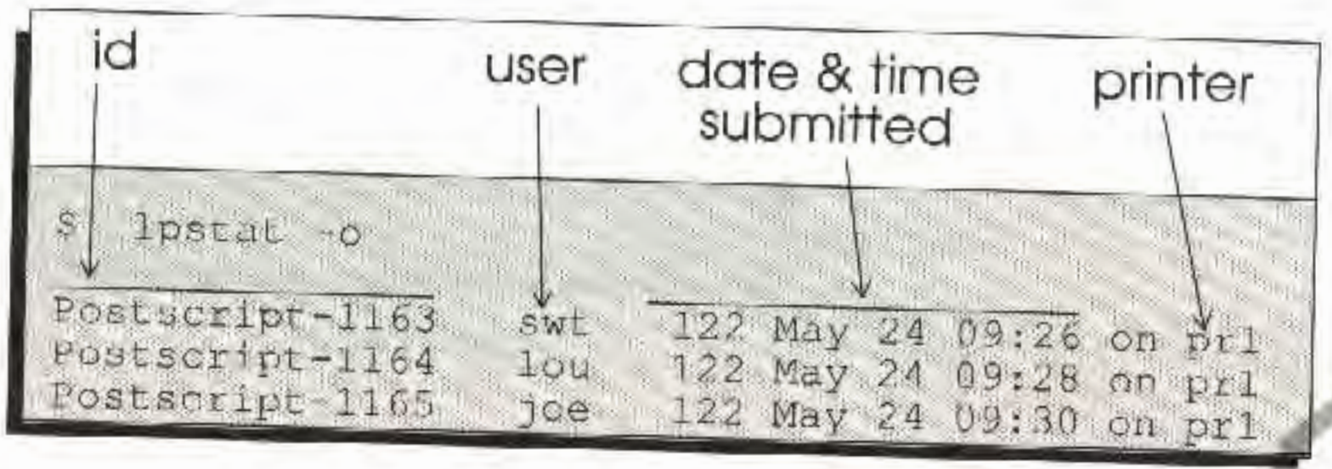
```
$ lp -o nobanner filename
```

When you issue the **lp** command, the file goes to a print queue where it waits its turn to print. You can check on print jobs that are waiting in the queue by using **lpstat**.

Checking on print jobs

Use lpstat and lpstat -o

lpstat shows only your print jobs. **lpstat -o** shows you all the print jobs in the queue, including the order in which they are scheduled to print.



lpstat options

Here are some useful options for **lpstat**:

Option	What does it do?
-u *username*	show print jobs for specific user
-t	show information about all printers
-r	tells you if lpsched is running
-v	show the device for this printer
-p name	show status for a specific printer
-d	identify the current default printer
-o	show print jobs for all users

Cancel a print job

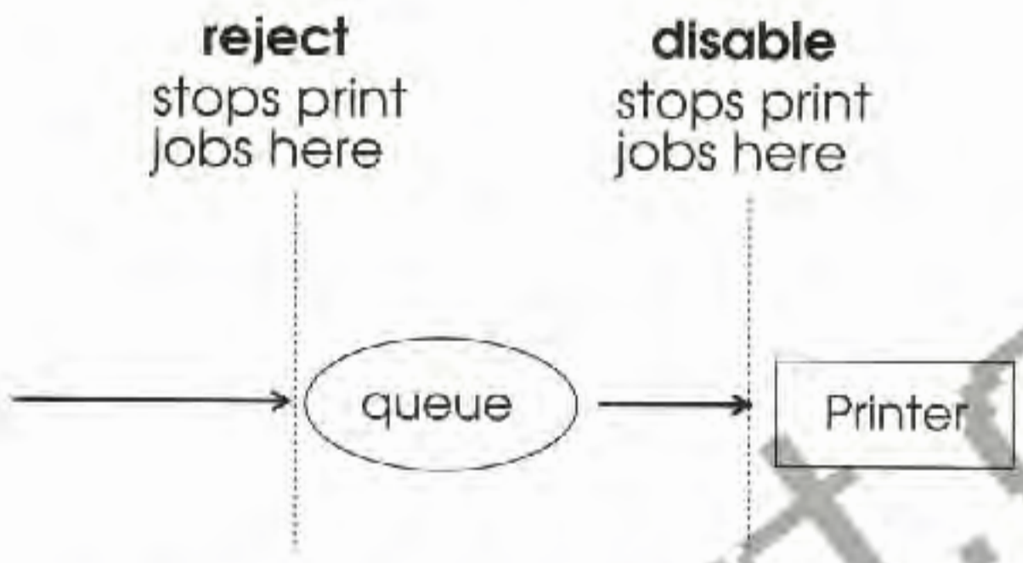
Use the **cancel** command to remove a print job from the queue. You can only cancel your own print jobs, unless you log in as root.

```
cancel id
```

Stopping the lp print service

Prevent users from printing

Two commands let you stop users from printing, **reject** and **disable**. You must log in as root or lp to use these commands, or to restart **lpsched**. **reject** prevents the print queue from accepting jobs. **disable** prevents the printer from printing jobs that are already in the queue.



When you disable the printer, **lpsched** will still queue your files, but they won't go to the printer until you enable printing again. If you reject the printer, **lpsched** will tell you that it can't print your file. **reject** prevents the queue from accepting any new print jobs. Use **disable** if you need to turn a printer off for a short period of time, but don't want to panic users who depend on the printer. Use **reject** if you want users to know that printing doesn't work. You can use **-r** ***message*** with **reject** and **disable**. Type **accept** or **reject -r** followed by a messge that lets users know why printing does not work.

If your file does not print:

Use the **enable** command to start sending print jobs to the printer again.

Use the **accept** command to start accepting new print jobs again.

If the file doesn't print, and you are sure your Amiga and printer are configured properly, **lpsched** might not be running. Check it using **lpstat -r**. To turn **lpsched** on, type **/usr/lib/lp/lpsched**. If **lpsched** fails to start or quits, you should check /var/spool/lp for a file called SCHEDLOCK. Remove this file then run **/usr/lib/lp/lpsched** again.

Also, type **accept *printername*** and **enable *printername*** to make sure the queue and printer are ready to process jobs, and make sure the default printer is the one you are trying to use.

Troubleshooting

Problem	Solution
Can't print	Use **lpstat** to see if the file is in the print queue. Check your mail. **lp** sends print errors to you through mail. **lpsched** is accepting print jobs, but not printing them. Type **enable** to start printing again.
Files aren't accepted for printing	Use **lpsched -r** to see if the scheduler is running. Type **lpsched** to turn the print scheduler on. Type **accept** to start accepting print jobs.
lpsched stopped working	Check /var/spool/lp for a file called SCHEDLOCK. Remove the file and restart **/usr/lib/lp/lpsched**.

Problem	Solution
Your PostScript file prints numbers instead of text	Tell the interface not to convert it. `lp -o postscript filename`
lpstat says it is printing, but nothing happens	Turn the printer off, wait a minute, turn it back on. **disable** the printer, **reject** it, turn it off, wait, turn it back on, **accept** it, and **enable** it. Cancel the print job and print it again.
lp is unable to read the file	Change the file permissions for your directory. **lp** can't print your file if "others" don't have execute permission in your current directory. Read your mail. It might have an error message that explains the problem

Using the vi editor

Special keys

ESC	cancel insert mode
CTRL-L	redraw screen
CTRL-D	scroll down
CTRL-U	scroll up

Insert and append

i	insert text
a	add text
o	add blank line

Quit and save

:q!	quit without saving
:wq	save and quit a file
:w	save your work

Change text

cw	change word
cc	erase a line and insert
C	change to end of line

Delete text

x	delete character
dw	delete word
dd	delete line
***n*dd**	delete *n* lines

Undo and repeat

u	undo last change
U	undo changes to line
. (dot)	repeat last change

Move and copy

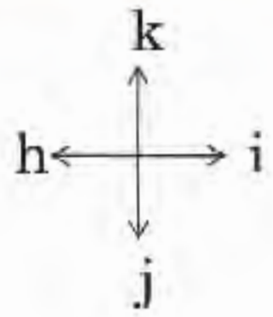
dd	cut
yy	yank (copy)
p	paste below

Search

/	search
n	repeat search
N	reverse search

Move in a file

w	to next word
0	back to beginning of line
$	to end of line
G	to the last line
:n	move to line number *n*



Using the vi editor

Why should you read this chapter?

This chapter provides a brief overview of basic **vi** (**vi**sual text editor) editing commands. To find out more about **vi**'s advanced commands, read the **vi man** pages or a **vi** reference book that you can find in your local bookstore.

This chapter shows you how to use **vi** to:

- open and view text files
- insert text
- move within a file
- change or delete characters, words, lines, and blocks of text
- correct mistakes by "undoing" them
- edit text
- repeat command sequences
- move and copy blocks of text
- search for a word or phrase

Special features

This section describes **vi**'s special features, including the fact that it is a modal program, doesn't automatically wrap text, and is case-sensitive.

Switch between modes

The **vi** editor has a unique editing feature; keys perform differently depending upon what "mode" you're in. **vi** works in three modes: basic command mode (keys perform editing tasks), file command mode (save and quit), and insert mode (keys insert text characters).

Use ESC to end insert mode

Use the ESC key to end insert mode and return to command mode.

When you first enter **vi**, you're in command mode. You can press an insert command key (such as **i**) to begin typing text. When you want to edit your work (to change, add, or delete a word), press ESC to end insert mode and return to command mode. While you're in command mode, the keys on your keyboard issue commands; you can't enter text until you press a insert command key (**a**, **A**, **i**, **I**, **o**, **O**).

If you forget which mode you're in, just press ESC to return to command mode, then start again.

Use RETURN key at the end of a line

The **vi** editor doesn't automatically wrap text, so you must press RETURN when you reach the end of a line on your screen.

Set wrapmargin

However, you can tell the editor to automatically insert RETURNs near the end of each line by typing **:set wm=10** (set wrapmargin to 10 characters from the right) while in command mode. The **:set** command doesn't reformat the rest of the paragraph; it simply breaks lines.

Colon commands

You also use RETURN to complete certain commands. Whenever you use a colon command (**:set**, **:w**, **:q**, **:wq**, **:q!**), you must press RETURN to execute it.

vi is case sensitive

Because some commands use the same letter, **vi** is case-sensitive. For example, if the instructions tell you to press **r**, press the key marked R. If the instructions tell you to press **R**, press the SHIFT key and the R key together.

CTRL-L to redraw screen

Use CTRL-L to redraw your screen if it becomes garbled. You may need to use CTRL-L if someone tries to use the **talk** command while you're typing, or if other system messages come through to your screen.

Starting and ending the vi editor

Start vi

To start the **vi** editor, type **vi** and the name of your file. If you don't include a filename, **vi** starts editing in an unnamed buffer; you can't save the file until you name it. (Refer to the *Troubleshooting* section for directions on saving an unnamed file.)

Save and quit commands

Here's a list of the save and quit commands. Remember to press RETURN after typing any colon command.

Command	What does it do?
:w	save your work
:q	quit **vi**
:wq	save your work and quit **vi**
:q!	quit without saving your work
:r *file*	read another file into your current file
ZZ	same as **:wq**; save and quit

You'll see the colon commands at the bottom of your screen, but don't worry; they won't appear in your text file.

Be careful with :q!

Be careful with the **:q!** (quit without saving) command; it quits **vi** without saving any changes you made to the file.

Save your work occasionally

Use the **:w** (write) command to save your work from time to time, just in case something happens to your system.

Moving around in a file

Why is your cursor important?

This section explains how to move your cursor around in a file. Moving the cursor is important because you need to put it in the right spot before performing a command or typing text. You can move it ahead or back by lines, words, or characters.

Cursor movement key commands

Here's a list of the most frequently used cursor movement keys. Remember to press ESC if you're in insert mode before using these commands or you will type the characters into your file.

Key	Movement
h	left
j	down
k	up
l	right
w	forward a word
b	back a word
0 (zero)	beginning of line
$ (dollar sign)	end of line
CTRL-D	scroll down
CTRL-U	scroll up
*n*G	*n*th line in the file (for example, 1G for the first line of the file)
G	last line in file

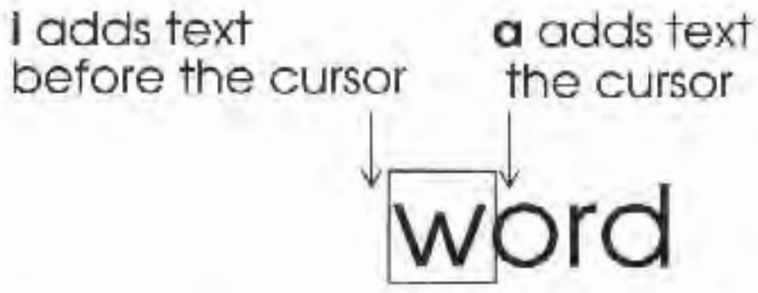
Using arrow keys

You can use the arrow keys to move the cursor around while staying in insert mode. You don't need to press ESC before using the arrow keys; they're not considered command keys, but rather special function keys within insert mode.

Adding text

Inserting and appending text

You add new text before the cursor when you "insert" text. When you "append" text, you add new text after the cursor.



Add text before or after the cursor

Insert and append commands

Here's a list of insert and append commands.

Command	What does it do?
a	add text after the cursor
A	add text at the end of a line
i	insert text before the cursor
I	insert text at the beginning of a line
o	insert a new line below the current line
O	insert a new line above the current line

Insert blank lines

To insert a blank line, press **o** (open a line); the blank line appears below the cursor line. Press **O** (SHIFT-O) to place the blank line above the cursor line.

Deleting text

Delete commands

You can delete single characters, words, lines, or blocks of text. Here's a list of delete commands.

Command	What does it do?
x	delete character
dw	delete word
dd	delete line
*n*dd	delete *n* lines (where *n* is a number)
D	delete to end of line
dL	delete to the bottom of the screen

Delete a single character (x)

To delete a single character, move the cursor to the character and press **x**.

Delete word (dw)

The **dw** (delete word) command deletes a word. Move the cursor to the beginning of a word and press **dw**. If you place the cursor in the middle of a word, the text from the cursor to the end of the word disappears.

If you accidentally place the cursor on the space before the word, the editor deletes the space first. You have to press **dw** again to delete the word. **vi** considers the space as a word.

Delete a line (dd)

Deleting a line is very simple. Place the cursor anywhere on the line you want to delete and press **dd**.

Delete several lines (*n*dd)

Use the ***n*dd** command to delete more than one line. For example, to delete a block of 20 lines, place the cursor on the first line of the block and press **20dd**. (**dd** without a number assumes you want to delete 1 line.)

Delete to end of line (D)

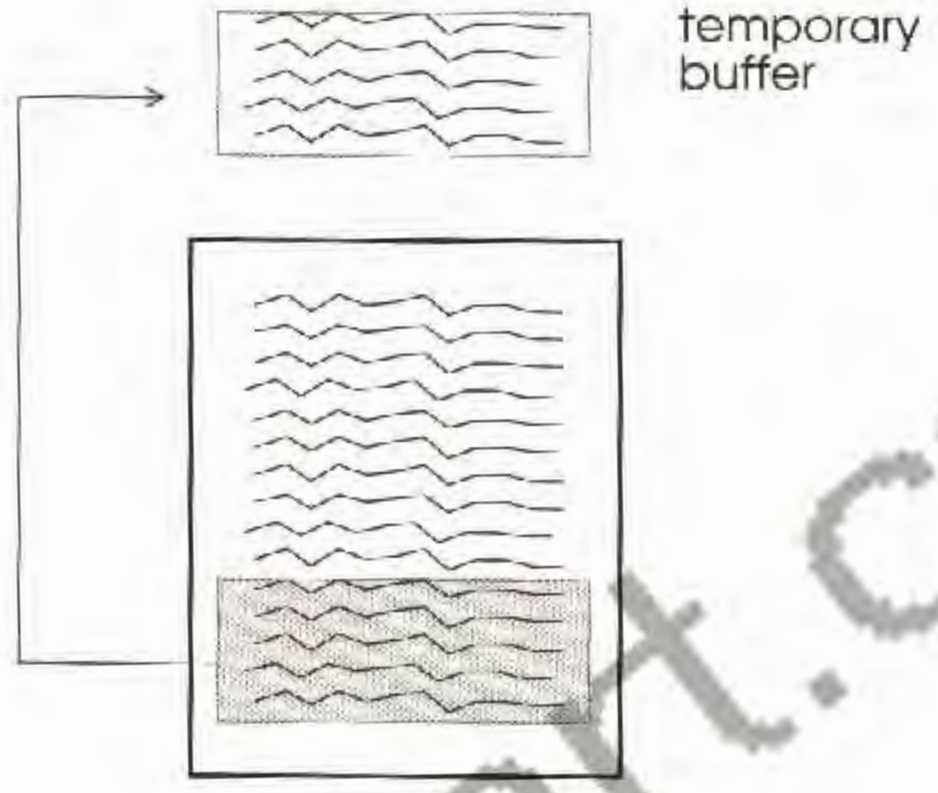
If you want to delete the rest of a line, press **D** (SHIFT-D). This command deletes text from the cursor to the end of the line.

Delete to end of screen

To delete the rest of the lines on your screen, move the cursor to the first line you want to delete and press **dL**. The editor deletes all the lines to the bottom of the screen and automatically moves the remaining lines in the file up to fill the current screen.

Temporary buffer

Any text you delete is put in a temporary holding space called a "buffer". You replace the buffer's contents each time you delete text.



Deleted text is put in a temporary buffer

Undo delete or paste text

You can recall deleted text through the undo and paste commands. Undo (**u**, **U**) puts the deleted text back in the file as if you never deleted it. You can place the deleted text anywhere in the file by moving the cursor and using a paste command (**p**, **P**).

Moving and copying text

When you move text, you delete (**dd**) it from its current location and then paste (**p**, **P**) it somewhere else in the file. Copying text (**yy**) puts a copy of the text in the buffer and leaves the original lines in place. You can then paste the copy anywhere in the file.

Moving and copying commands

Here's a brief list of moving and copying commands:

Command	What does it do?
J	join lines together
yy	copy ("yank") a line
*n*yy	copy *n* lines (where n is a number)
dd	cut (delete) a line
*n*dd	cut (delete) *n* lines (where *n* is a number)
p	paste a line below the cursor line
P	paste above the cursor line

Join lines (J)

The **J** command reformats text. As you delete or edit text, you may produce a lot of short lines.

```
The sky
is grey
and cloudy.
```

Using J to make lines look better

You can join lines with the **J** command to make them a more reasonable length.

```
The sky is grey and cloudy.
```

vi expects you to use the **J** command or a separate formatting program (such as **nroff**) to format lines and paragraphs before you print the file.

Copy line (yy)

To copy a line of text, move the cursor anywhere on the line you want to copy and press **yy** (yank). You don't see anything happen on your screen; however, the editor has placed the line in a temporary buffer.

Paste lines (p or P)

Use one of the paste commands (**p** or **P**) to duplicate the line somewhere else.

Copy several lines (*n*yy)

Use the ***n*yy** command to copy several lines of text. For example, to copy 20 lines of text, move the cursor to the first line of the block and press **20yy**. Then, move the cursor to where you want the text, and press a paste command (**p** or **P**).

Special buffers

Actually, the temporary buffer's contents are not really deleted; they are actually moved to a "history buffer". Read an advanced **vi** manual to learn about these and other special buffers.

Changing text

Change commands

To change text, move the cursor to where you want to make your change, press the appropriate command (**cc, C, cw**, **r**, **R**), and rekey the text.

Here's a list of the most frequently used change commands.

Command	What docs it do?
cw	change a word
cc	erase a line and insert
C	change to end of line
r	replace a single character; remain in command mode
R	retype to the end of a linc; remain in insert mode

Replace a character

You can change a single letter with the **r** (replace) command. Simply move the cursor over the letter you want to change, press **r**, then type the correct letter. After you replace the character, **vi** is in command mode.

Change word (cw)

To change a single word, move the cursor to thc beginning of the word you want to change and press **cw** (change word); **vi** places a $ (dollar sign) at the end of the word. Type the correct word or words over the wrong word. Press ESC when you complete the change.

Punctuation is considered a word

The **cw** command considers any punctuation mark as a separate word. For example, if you want to change a word in quotes, move the cursor to the first letter of the word, not to the quotation mark.

vi has several ways to change a line: **cc**, **C**, and **R**.

Change an entire line (cc)

If you want to change an entire line, use the **cc** command. **cc** erases the current line and puts you in insert mode so you can rekey it.

Change to the end of a line (C)

If you only want to change the end of a line, use the **C** (SHIFT-C) command. This command erases text from the cursor to the end of the line and puts you in insert mode so you can rekey the text.

Replace more than 1 word (R)

Another way to change the text on a line is the **R** (replace) command. **R** puts you in a replace mode. Each character you type replaces the current character on the line. When you finish a line, you're still in insert mode. Any new lines you type are added. **R** replaces only the current line; it doesn't continue in replace mode after that line, although you're still in insert mode and can keep adding new lines.

Undoing changes

This section explains how to undo changes and correct editing errors with the undo commands.

Undo commands (u, U)

Command	What does it do?
u	undo the last command
U	undo all the changes recently made to the current line

Use to fix accidental deletes and cuts

The undo commands are useful for:

- bringing back text you accidentally delete
- removing text you accidentally insert
- changing your mind about an edit

Temporary buffer

Remember, deleted text is placed in a temporary buffer so when you undo the delete commands, the editor pulls the text out of the buffer and puts it back into your text file.

Difference between u and U

The difference between the **u** and **U** commands is the number of changes that are undone. The **u** command undoes only the last command. The **U** command undoes all the commands performed on the current line since you moved to it. However, if you move off a line, **U** can no longer undo changes to it.

Searching for text

Search commands

You can look for a word or phrase with search commands. The search commands look forward (**n**) or backward (**N**) for one or more characters. Here's a list of the most frequently used search commands.

Command	What does it do?
/ (slash)	search for one or more characters
n	repeat search
N	reverse search

While in command mode, press /, type the text you're looking for (called the "search string" because it's a string of characters), and press RETURN. **vi** puts the cursor on the first match; if necessary, the editor scrolls through the file to find the string of characters.

Search (/) and next (n, N)

Press **n** (next forward) or **N** (next backward) to search for the next occurrence of the string.

Use search with change and repeat commands

The search command is frequently used in conjunction with the change and repeat commands. You can use search and repeat commands to move through a file and change all or some occurrences of a specific string.

vi also has many powerful replace functions, some of which work in conjunction with the search command. Read any advanced **vi** manual to learn about the replace functions.

Repeating changes

Repeat changes (.)

Use the dot (period key) command to repeat the last change you made. This is a very useful function when you want to repeat a command without pressing the command sequence again.

Use with other commands

You can use the dot command to repeat the following commands:

- delete (**dw**, **D**, **dd**, ***n*dd**)
- insert (**i**, **I**, **a**, **A**, **o**, **O**)
- change (**r**, **cw**, **cc**, **C**, **R**)
- copy and paste (**yy** or ***n*yy** and **p** or **P**)

NOTE: The dot command doesn't work with searches; to repeat a search, use **n** or **N**.

Repeat delete

For example, to delete words on different lines, simply press **dw** for the first delete, then move to the next word you want to delete and press the period key. The dot command saves you from having to press **dw** for each word you delete. The dot command works the same way with any other command.

Use with search and replace

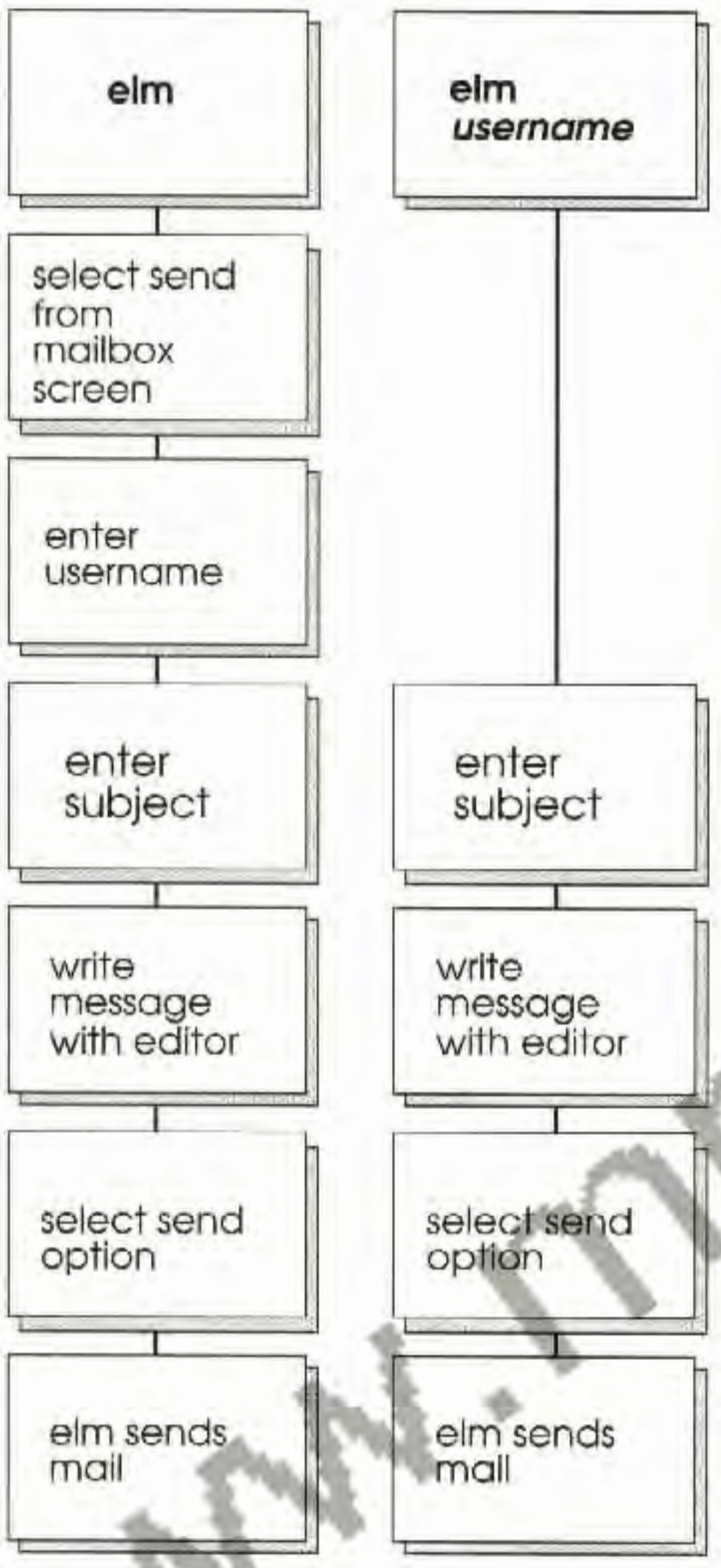
Use the dot command with search and replace sequences. For example, you can easily change a word throughout a file. First, look for the word with the / (slash) command, then change it with the **cw** command and press ESC. To continue through the file, press **n** to find the next occurrence and press the period key to repeat the change.

Troubleshooting

Problem	Solution
Your command doesn't work and appears in your text file	You're probably not in command mode. Press ESC, then press the command key.
Your text doesn't display on your screen	You're probably not in insert mode. Press ESC and one of the text insertion keys (**a**, **A**, **o**, **O**, **i**, **I**).
You can't save your file or exit vi	You probably entered **vi** without a filename. You must save your work to a file by using the **:w** command. Type **:w** ***filename*** and press RETURN.
The wrong file or a blank file appears on your screen	You probably typed the wrong filename. Quit out of the current file with **:q!** and retype **vi** ***filename*** with the correct filename. **vi** is case-sensitive and considers a blank between two words as two separate file names, so make sure you type the filename properly.

Using electronic mail

Ways to use elm



Read message menu

?	help
b	bounce (forward)
d	delete
g	group reply
i	return to mailbox
p	print
Q	quit immediately
r	reply
s	save in file
u	undelete
SPACE	read next message

Mailbox screen options

?	help
c	change folders
d	delete
m	mail a new message
o	options
p	print
q	quit
r	reply
s	save in file
u	undelete
x	quick quit

Using electronic mail

Why read this chapter?

This chapter gives you a brief introduction to electronic mail. You'll learn how to use **elm** and **mail** to send, read, delete, and store messages.

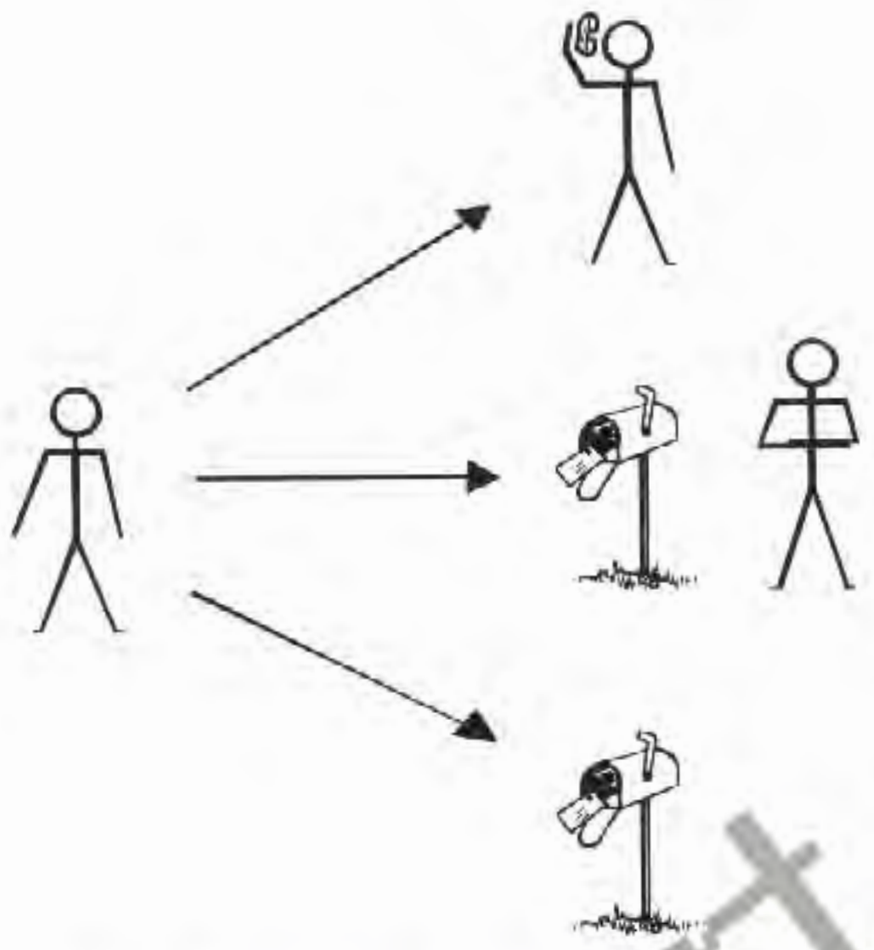
Why use electronic mail?

Electronic mail is a quick and easy way to communicate with other users. You can use electronic mail to send messages or files to co-workers. You can use it to ask questions and send replies. It works even if other users aren't logged in. Mail messages wait in their mailboxes until they're ready to read them.

Electronic mail is more than a method of sending messages; it's also a means of managing mailed files. You can create and send a memo to a group of people, then print it, delete it, or store it in a folder so you can refer to the information later. You can also use mail to send notes to yourself.

Comparing talk to electronic mail

Unlike the **talk** program, recipients of mail don't have to be active users at the time someone sends them a message. You can send mail messages to users at any time and your mail will wait for the recipients. As soon as they log in to the system, they'll see a message that says "you have mail." Then they can use one of the electronic mail programs to read and respond to the message.



You can talk only to active users, but you can send mail to active and inactive users

Types of electronic mail

You can use several different types of electronic mail with UNIX. Two of the most common programs are **mail** and **elm**.

What is elm?

The **elm** program is menu driven and provides help messages and prompts on-screen as you work. You can set **elm** menus to match your level of expertise through the **elm** options. This chapter briefly describes the options available for beginners.

What is mail?

You can also use **mail**, the traditional UNIX mailer, to send and read messages; however, the **mail** program doesn't include a menu and doesn't have all the extra features that **elm** has. It is most useful for sending quick, short notes.

Starting elm for the first time

Create elm directory and mail folder

The first time you try to use **elm**, the program displays several prompts that help you automatically set up **elm** directories on your system.

To begin the initialization, type **elm** at your shell prompt. Two notices appear asking if you want **elm** to set up an **elm** directory and a mail folder for you. Answer yes (press **y**) to both these queries. **elm** then creates the Mail directory and displays the mailbox screen.

Sending mail through elm

Mailbox screen

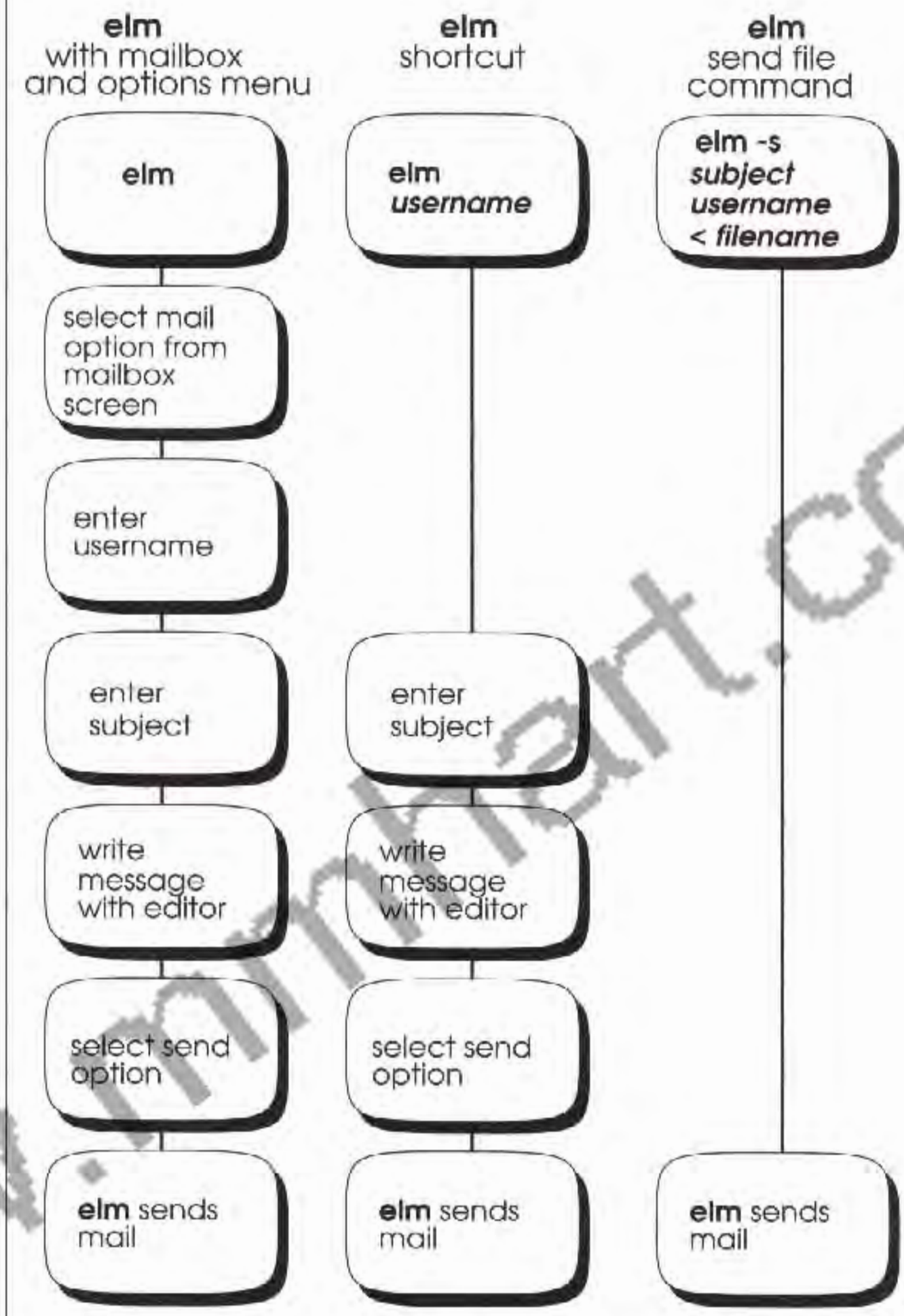
Shortcut

Mail files

There are three ways to send mail through **elm**:

- type **elm** at a shell prompt to start the **elm** mailbox screen
- type **elm** ***username*** to skip over the **elm** mailbox screen and go directly to the subject of your message
- type **elm -s** ***subject username < filename*** to mail a file to another user

3 ways to send mail



Three ways to send mail

Send command

The simplest way to send a message is by typing **elm** ***username*** at the shell prompt.

```
$ elm joe
```

Enter subject and copy to list

elm displays the full name of the user and prompts you for the subject and the *usernames* of people who should receive copies of the message.

```
To:  Joe Smith
Subject of message: Lunch
Copies to: carol
```

Use a text editor to write the message

elm starts the text editor (**vi** is the default editor) so you can type the message. When you have saved your message and quit from the editor, **elm** asks you to send, cancel (forget), or edit the message, or change the headers.

```
Please choose one of the following options by
parenthesized letter: s
 e)dit message, edit h)eaders, s)end it, or
 f)orget it.
```

Send message

To send the message, simply press RETURN; send (s) is the default value.

Forget message

To forget (cancel) a message, type **f**. The message is stored in a file (cancelled.mail) in your home directory.

Change message

If you want to change or expand your message, you can edit it by pressing **e**. **vi** displays your message file so you can edit it. After you save your corrected message, you can send it.

Change header information

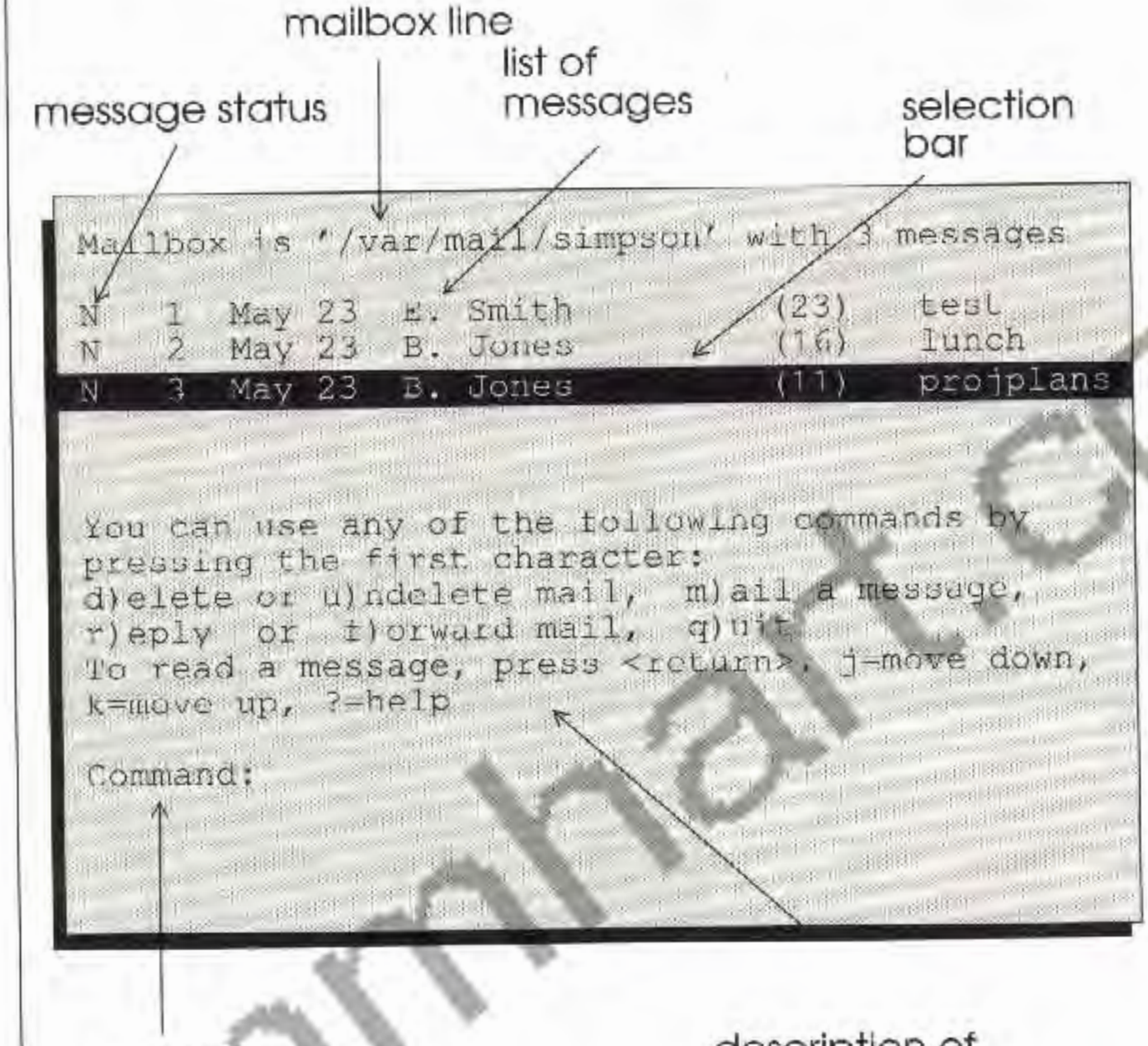
If you want to change or add information to your headers, press **h**. You can change the subject, the recipient, and other types of address information.

```
        Message Header Edit Screen
To: joe@gallium (Joe Smith)
Cc:
Bcc:
Subject: Staff memo

Reply-to:
Action:
Expires:
Priority:

Choose first letter of existing header,
u)ser defined header, or <return>.
Choice:
```

Reading mail through elm

Access elm mailbox screen

Type **elm**; the first screen you see is your mailbox, which includes an index of mail messages and a menu of options.



Sample of the elm mailbox screen

Message lines

The selected message line is highlighted; whatever command you choose affects the highlighted message. You can select any of the messages on the screen by pressing **j** to move down the list of messages or pressing **k** to move up. Press SPACEBAR to read a message.

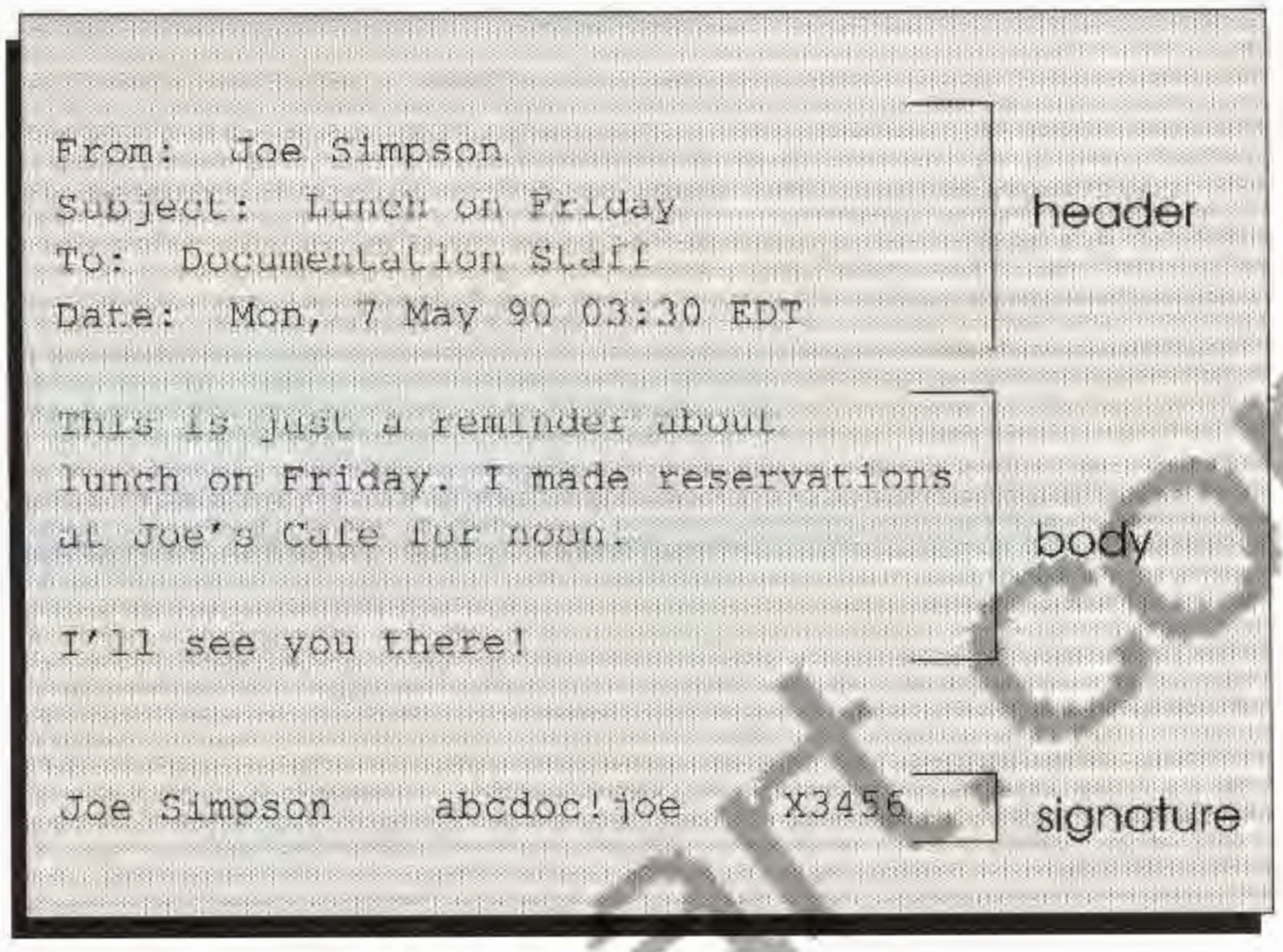
```
Mailbox is '/var/mail/simpson' with 3 messages

N   1  May 23  E. Smith          (23)   test
N   2  May 23  B. Jones          (16)   lunch
N   3  May 20  B. Jones          (22)   testplans
```

move bar	read messages	delete message	reply to message
j or **k**	SPACEBAR	**d**	**r**

Using the elm mailbox screen

Parts of a message

A message has three parts: a header, a body, and a signature.



Sample message

If the message contains more lines than can fit on a screen, press SPACEBAR to scroll to the next page of the message.

If you have more than one message to view, press the SPACEBAR to display the next message.

After you read a message...

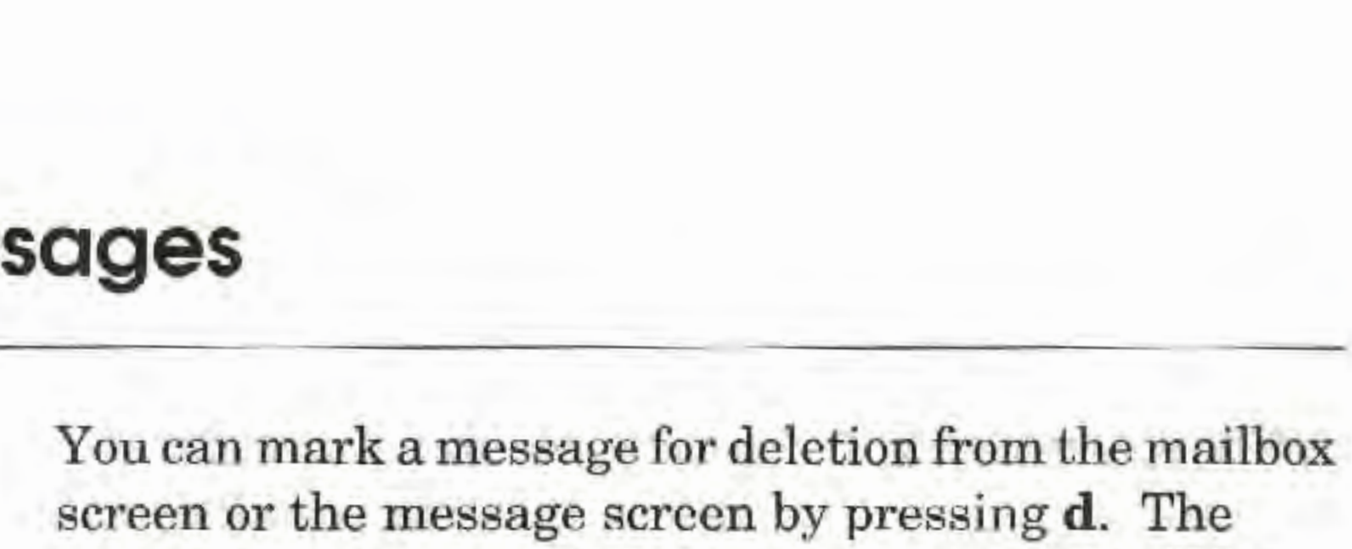
After you read a message, enter an option at the command prompt. Here's a partial list of options; notice that some don't appear in the menu at the bottom of the screen, but are listed in the on-line help screen.

Command	Key	Description
copy	C	copy the message to a designated folder
delete	d	delete message
help	?	access information about commands
group reply	g	reply to everyone who received the message
print	p	print the message
bounce	b	bounce (remail) the message to another user
reply	r	reply to the message
save	s	save the message in a folder
undelete	u	undo the delete request for a message
index	i	return to the mailbox screen
quit	q	quit elm

Deleting messages

Mark message for deletion

You can mark a message for deletion from the mailbox screen or the message screen by pressing **d**. The messages appear on the mailbox screen with "D" as the status. This means that you've selected the messages to be deleted; however, **elm** hasn't deleted them yet. Messages aren't deleted until you quit out of **elm** and confirm that the messages should be deleted.



Quit

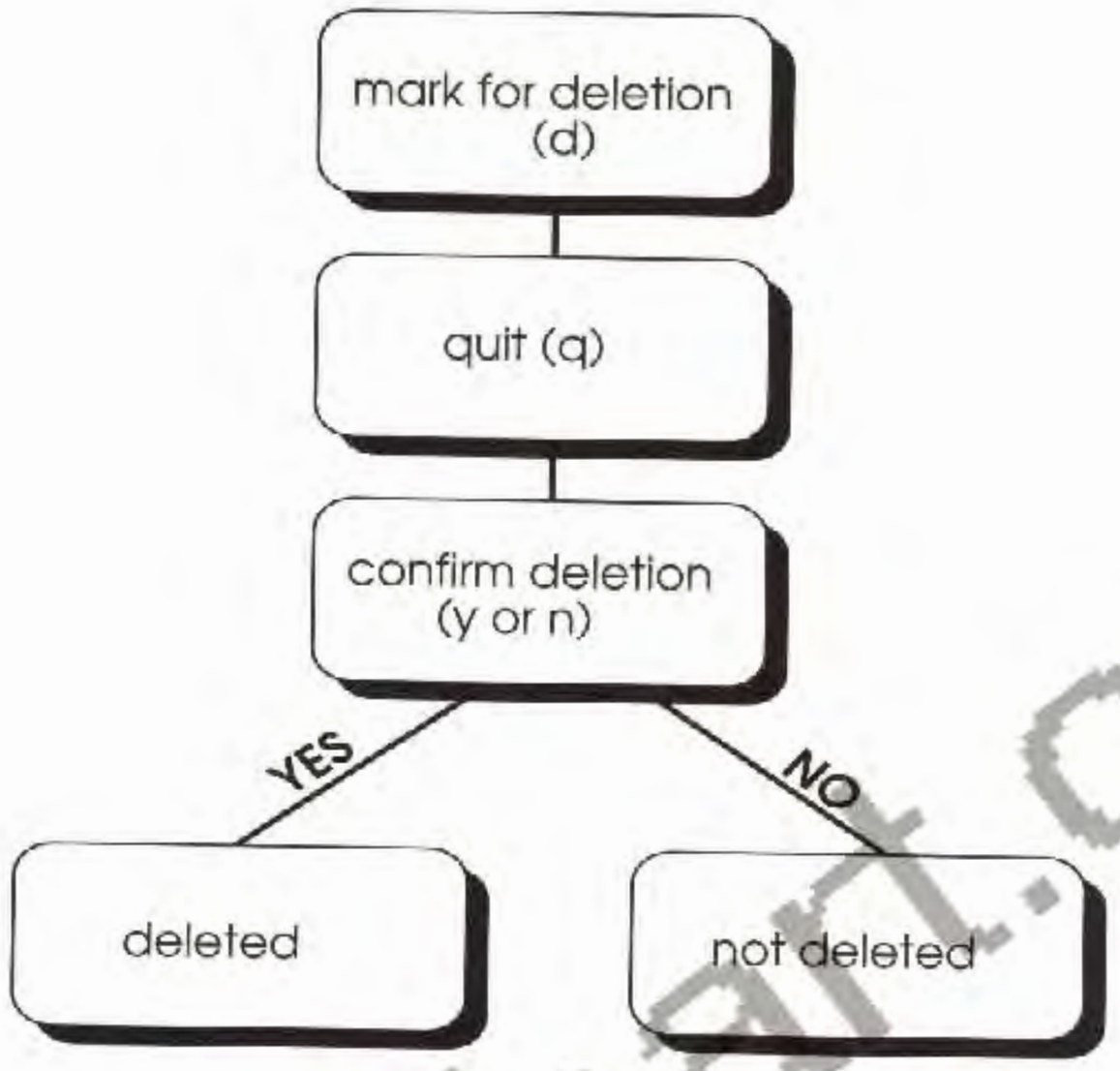
When you press **q** (to quit), **elm** displays a confirmation message.

Confirm the deletion

Press **y** to delete the messages.

However, if you've changed your mind about deleting the messages, press RETURN for no (N is the default). The **elm** program won't delete the messages.

Steps for deleting a message

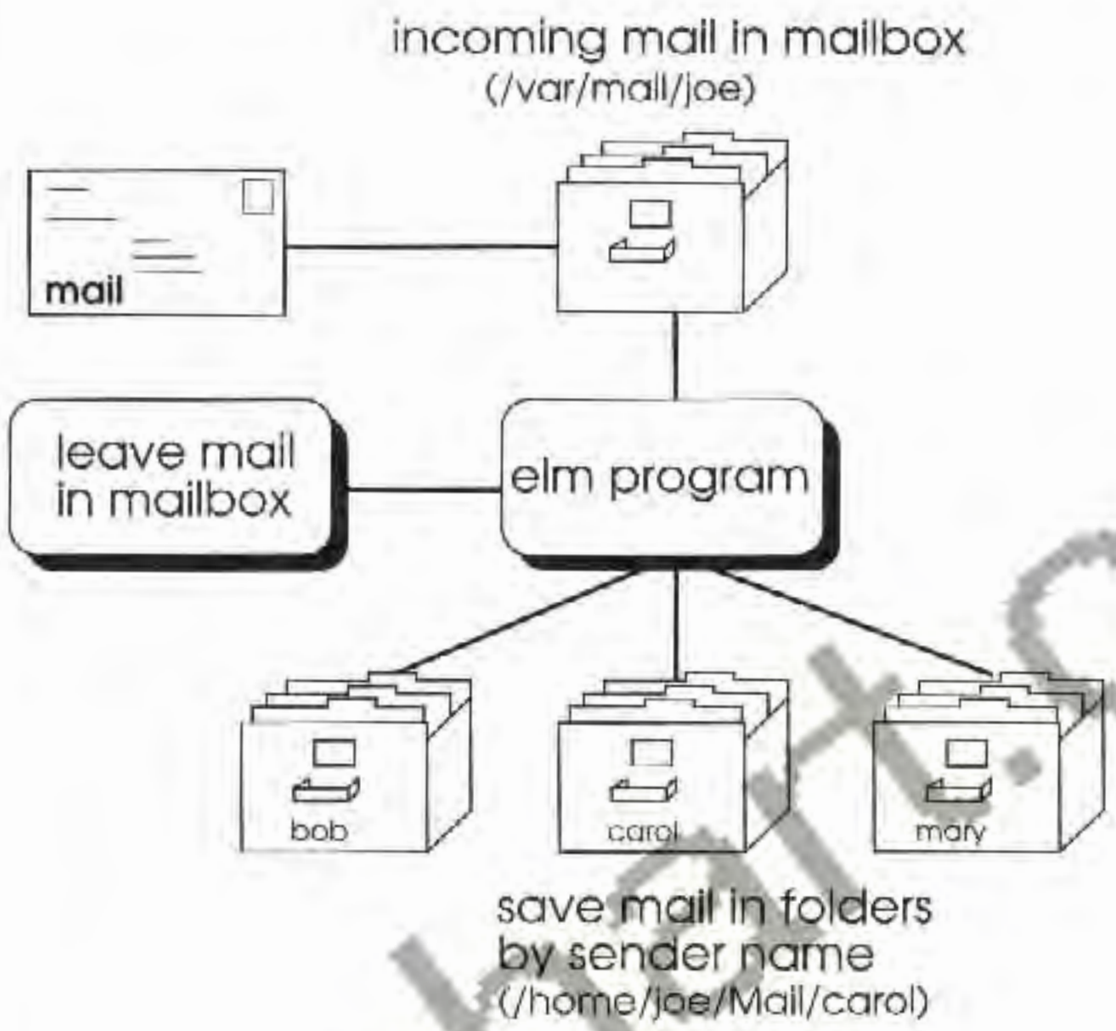


Several steps to delete a message

Saving and storing mail

Saving and storing mail

If you choose to save your mail messages, you can store them in a mail folder or leave them in your mailbox.



Saving mail messages

Save mail in sender folder

To save mail in a sender folder (/home/*username*/Mail/*sender*), press **s** at the command prompt on the mailbox screen.

Quit elm

When you use the **q** option to quit, **elm** displays a second confirmation prompt. If you answer yes, all the mail you've read is moved to a received mail folder (/home/*username*/Mail/received), where you can find it later. The default is no (which keeps the messages in your mailbox until you delete them).

```
Command: Quit     Store read messages in "received"
                        folder? (y/n) n
                          [Keeping messages]
$
```

Customizing your signature

In the *Reading a message* section of this chapter, you saw how each message begins with a header and ends with a signature line. This section explains how you can create your own signature line.

Edit signature files

To design special signatures, create two .signature files in your home directory. Then add the filenames to the localsignature and remotesignature fields in the .elm/elmrc file.

If you don't have an .elm/elmrc file, press **o** (for the options command) on the **elm** mailbox screen. Once you've accessed the options submenu, **elm** automatically creates an elmrc file for you in the .elm directory.

```
Let's have a planning meeting                ] Message
on Friday morning.

Joe Simpson  ABC Corporation  X3654           ] Signature
```

Local and remote signatures

If you like, you can use the same .signature file for both remote and local signatures. However, a local signature usually includes your *username* (joe), and a remote signature includes additional information that people outside your company need in order to contact you (full phone number, system name, e-mail address).

Customizing elm

2 ways to customize elm

You can customize the **elm** program in two ways: through the **elm** options screen and by editing the .elm/elmrc file.

Options screen

To access the **elm** options screen, type **o** at the command prompt on the mailbox screen. Here are brief descriptions of some options you can change.

Command	Option	Description
arrow	A	arrows or a reverse video bar as selection indicator
calendar	C	file where calendar entries are saved
display	D	program used to display messages
editor	E	text editor
folder	F	folder directory
menu	M	display commands on mailbox screen
outbound	O	file where copies of messages you send are automatically saved
print	P	printer where messages are printed
sort	S	order in which to display mail messages
user level	U	level of elm to use for menus and commands
your name	Y	the full name, comment, or group you want to display on messages

To find out more about these options, press **?** at the command prompt on the option screen.

Configuration file

Regardless of how you change options, **elm** automatically creates a configuration file named .elm/elmrc.

The file contains some of the same options as the options screen plus many others; it also provides simple instructions to help you edit the file. Here's a sample of an .elm/elmrc file.

```
#   .elm/elmrc - options file for the mail system

# For yes/no settings with ?, ON means yes, OFF
# means no

# where to save calendar entries
calendar = /home/joe/calendar

# what editor to use
editor = vi

# should the default be to delete
# messages we've marked for
# deletion?
escape = ~

# the full user name for outbound mail
fullname = Joe Smith

# where to save my mail to, default directory is
"Mail"
```

Sample .elm/elmrc file

Using mail

Differences between elm and mail

mail is another electronic mail program. You can perform many of the same tasks with mail as you can with **elm**; however, you cannot correct errors as easily with **mail**. You can't select menu options (although you can access a list of commands through help), and you can't select specific messages to read.

Send a message

To send a message, type **mail** ***username***. When the shell prompt disappears, type your message. End the message with a period (.) on a blank line; this tells **mail** to send the message.

```
$ mail joe
Here's a message
.
$
```

Read a message

To read your messages, simply type **mail**. The **mail** program presents all your messages, beginning with the last one received. Press RETURN to view each message.

After reading a message...

After reading each message, tell the **mail** program what to do with it. For a full list of **mail** options, press **?** at the prompt. Here are some of the options you can use.

Key	Description
d	delete
p	print
q	quit mail
r	reply to message
s *file*	save message in mailbox format
u *n*	undelete message number *n*
w	write as text file
x	quit mail without updating messages
RETURN	read next message

Troubleshooting

Problem	Solution
Can't start elm	You probably don't have .elm or Mail directories in your home directory. When you first try to use **elm**, queries ask if you want **elm** to create these directories. Make sure you answer yes.
Your elm commands don't work	You probably didn't type the command correctly. Check the screen prompts for messages and valid options. Make sure you include a valid *username* in the command line if you are using the **elm** shortcut. If you are sending a file through **elm**, make sure you include a **<** (less than sign) and a valid *username* and *filename*.
You receive mail intended for another user	Use either the **f** (forward) or **b** (bounce) command to send the message to the appropriate user.
Your send or copy list is incorrect or incomplete	If you haven't sent the message yet, you can use the **h** (edit header) command from the send message screen to make the necessary changes.

Problem	Solution
You decide not to send a message	If you have not sent the message yet, you can use the **f** (forget) command from the send message screen to cancel the message.
You accidentally mark a message for deletion	If you've marked a message for deletion but have not yet exited from **elm**, you can undelete the message with the **u** command or answer no to the "Delete messages?" query that appears when you quit **elm**.

Networking

Networking terms and concepts

Local world	The machine you are logged in to and using
Global world	All systems and processes on your network
Remote login	Log in to a different computer over the network by typing your login name and password, then use that system's resources; your local world is now the other system

Local tasks

who	List users who are logged in to the system
finger	Similar to who but more detailed
ps	Processes running on local machine
wall	Send a message to everyone logged in to the machine
kill	Stop a running process
date	Display the system's date and time
ls, cp	List or use local files

Global tasks

mesg	mesg y or n to receive or reject incoming talk
talk	Establish a dialogue with another user at any system on the network
mail	Send a message to another user's mailbox, where it waits until it is read
ping	Check if a system is on and connected to the network
rlogin	Log in to a different system
rcp	Copy files over the network to and from other systems

Networking

What is a network?

A network (in its most basic form) consists of computers connected together by a cable. Depending on the type of network, users of these computers can talk to each other, share files, exchange mail, and share peripherals.

What do you need to make a network?

One unique feature of UNIX is its built-in networking capabilities. With most operating systems, you have to buy special applications to share files and send mail. UNIX has its own mail system, its own system for sharing files over the network, and its own set of networking utilities. In most cases, UNIX has several different types of networking software, so you can connect to almost any kind of network.

Networks usually require some special hardware, such as an expansion card and a cable. Once you add these parts to your computer, you have everything you need to communicate over a network.

Local vs. global

There are two types of network activity: local and global.

Your local world is any system you happen to be logged into and any users logged into that same system. Your global world is every system connected to the network.

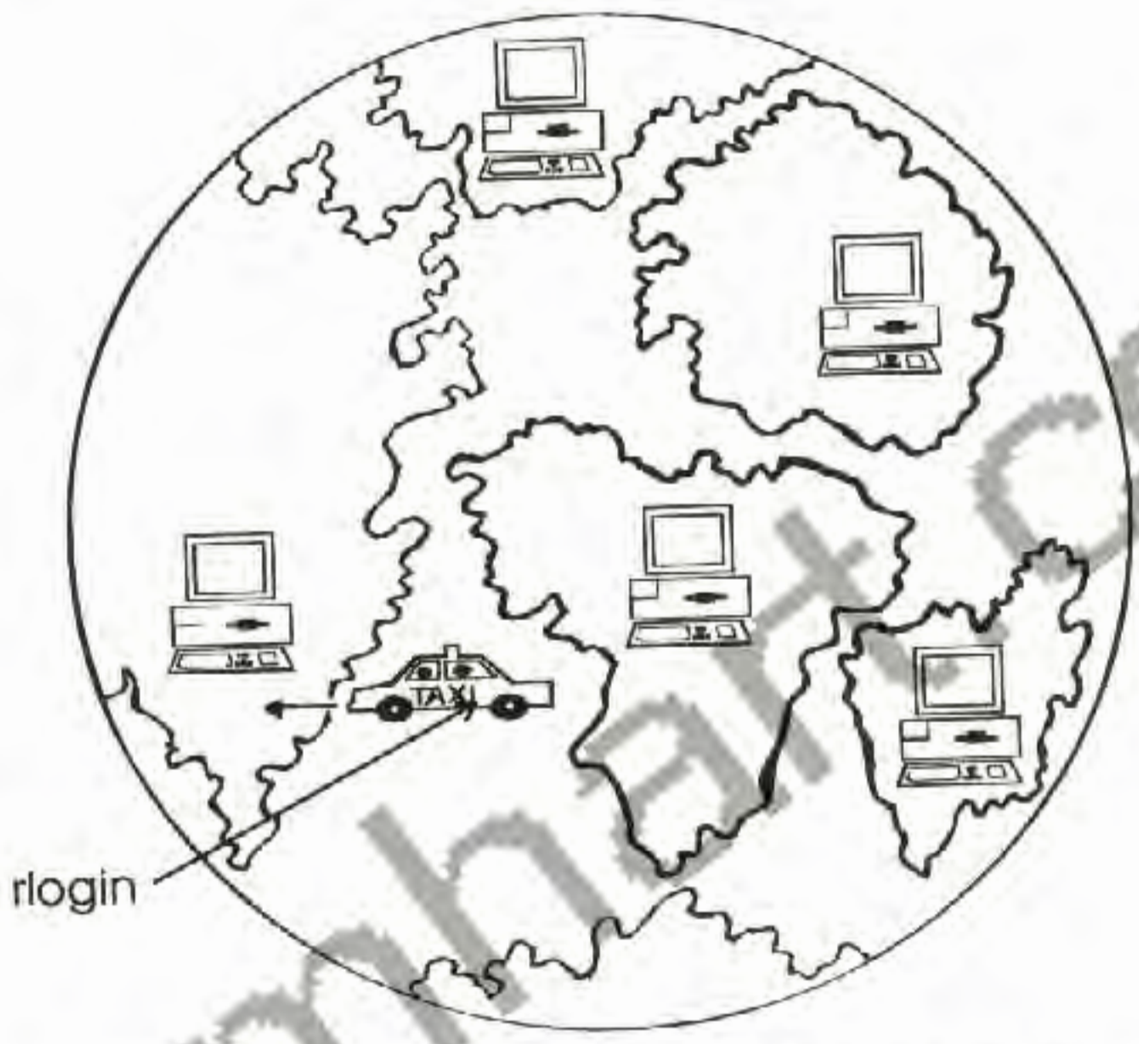
Local activity is relative to you, and takes place on the same system you're logged in to; global activity takes place all over the network around you. You can generally monitor local activity, since it takes place on the system you are using; it is very difficult to monitor or even guess at all the global activity that might be taking place on other network systems.

This chapter shows you how to perform local tasks and interact with a global community.

Working locally vs. working globally

What is a local environment?

Your local environment is any system that you are currently logged in to. This could be the system on your desktop, the mainframe computer in another building, or a different workstation to which you have "transferred" your session activity.



rlogin transfers you to a different system

How can you consider another system local? The answer is fairly straightforward. You can work on files, look at directories, work in your home directory, and so on when you log into your personal system. You can also log into someone else's system (if you have permission) and do these exact same things. The only difference is which system is processing all the work. When you are logged in to your personal

system, it's doing the work. When you log in to another system, that system is doing all the work and defines your local environment.

Local is a relative term

Your local environment is relative to where you are. Everbody has their own local world, which may or may not be the same as yours. You all share the same global world, but from your own local position. UNIX provides you with certain commands for getting information about your local environment.

Many different kinds of systems can be part of your global network; each system is still its own separate local world.

Global environment includes different types of systems

Local tasks

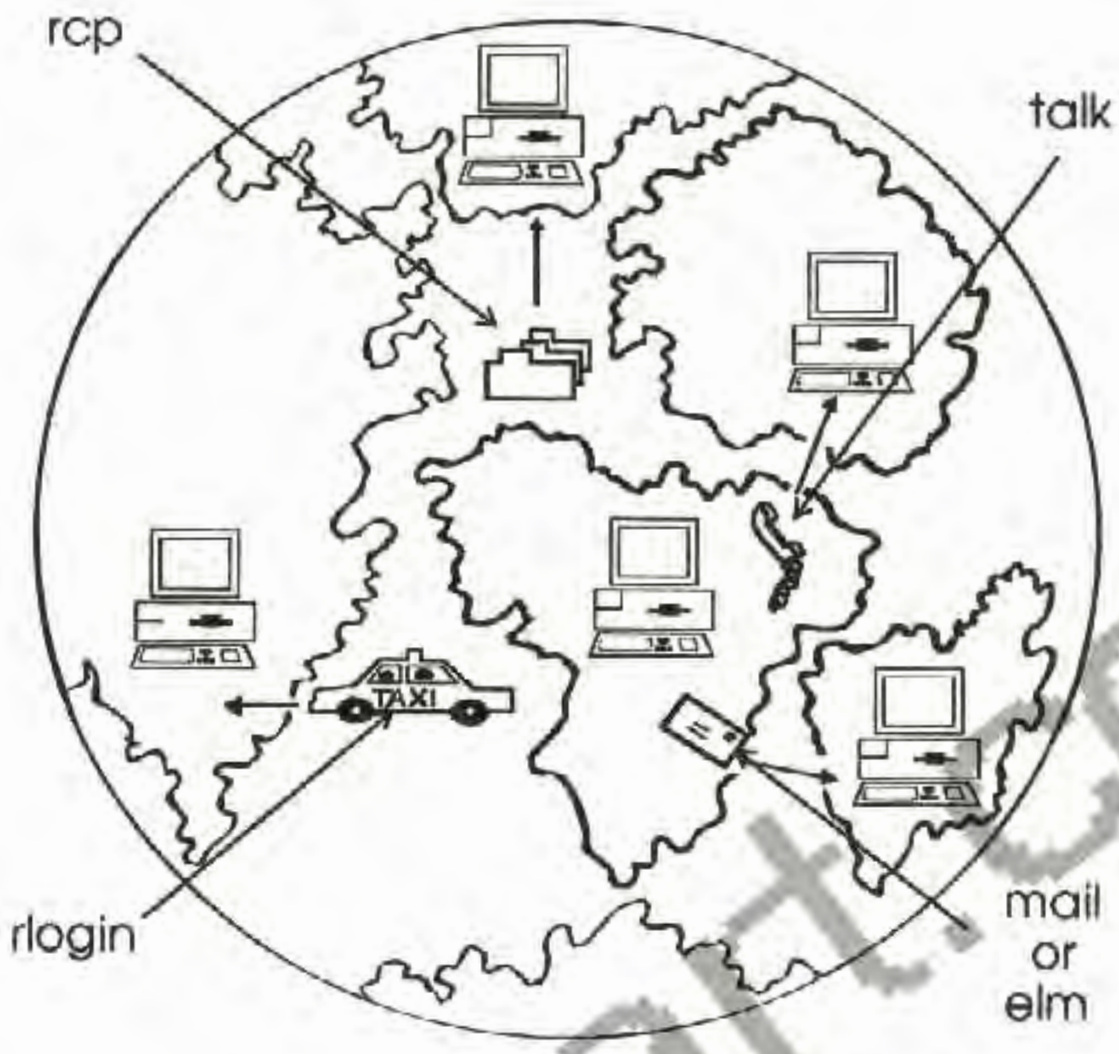
Local tasks include:

- listing processes
- listing current users
- working with files and directories
- broadcasting a message
- system maintenance

Global tasks

Global tasks are specific tasks you can perform that affect other users and systems on the network, either in the same building or perhaps many miles away. Some examples of global tasks are:

- having a conversation with another user over the network
- sending mail to users on other systems
- copying files to or from other systems
- changing your local environment to another system by remotely logging in over a network
- checking the status of another system
- network maintenance



Types of global network activity

Unlike your local environment, it doesn't matter where your global environment originates. You can perform global tasks, like talking over the network, no matter where you are logged in. The key to a distributed network is that some activity occurs everywhere, and no one system does all the work for everyone.

Looking at your local world

Check to see who's logged in

The **who** command lists all users currently logged into your system. The list you get depends on your local environment. If you are logged into your personal system, **who** shows you everyone else logged into your personal system.

If you are logged into another system, **who** lists the other users logged into that system.

Using the **who** command is simple.

```
$  who
joe      term/con2     May2 11:35
pat      term/con3     May3 08:34
```

The **who** command tells you that two users are logged in (joe and pat), what terminal they are logged in to, and the time they logged in.

Another method for seeing who's logged in

You can also use a command called **finger** to see who is logged into your local environment. **finger** is similar to **who**, except it gives you some more information about each user.

```
$  finger
Login     Name      TTY       Idle      When     Where
joe         Joe Jones  term/con2  4:01        Mon 07:29
```

In addition to login name, terminal, and login date, **finger** gives you the user's full name, how long he has been idle, and what he is doing.

A process starts when you type a command. The **who** command starts a process that looks for people logged into the system, then displays information about them. **ls** starts a process that displays the files in a directory. Applications such as word processors are also considered processes; the process is the entire program. Every UNIX command you type starts a process.

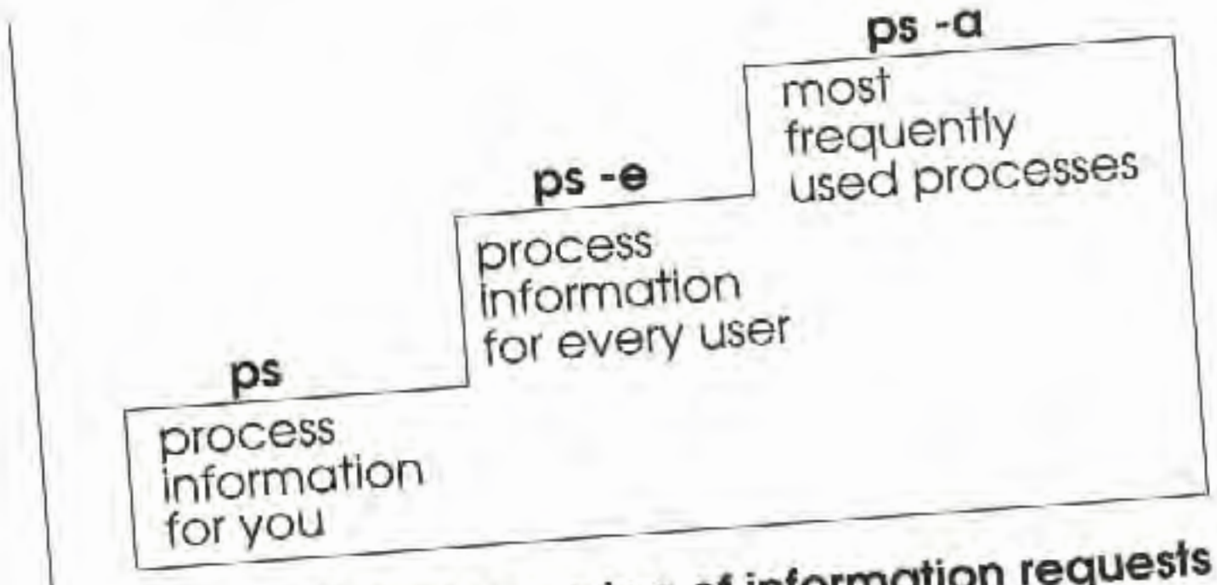
Check on running processes

You can check to see what processes are running on your local system by using the **ps** command with the **-e** option (for everyone).

```
$  ps -e
PID      TTY          TIME      COMD
138      term/con4    0:04      ksh
415      pts/5        0:09      rlogin
416      term/con4    0:00      ps
```

Information	What does it mean?
PID	ID number for this process
TTY	Terminal on which the process is running
TIME	Time the process has been running
COMD	Name of the process

ps has many more options. You can type **ps -a** and see only the most important user processes or you can type **ps -ef** and see all processes displayed in long format. See the man page for a complete list of **ps** options.



The progression of information requests

Broadcast a message to everyone on your system

Amiga UNIX lets you send a message to everyone who is logged into your computer. This is especially useful if you plan to shut your computer down and want to warn people using your system. The command is called **wall** (write to all users) and broadcasts a message to everyone logged into your system. Some users might have messages turned off, so you can't talk to them or send a message to them. Login as root before using the **wall** command to be sure that everyone gets your message.

```
$ wall
System is going down soon
CTRL-D
```

Type the message, then press CTRL-D to signal the end of the broadcast. Your message goes out to all users logged in to your system.

Working in a global world

Do you want other users to talk with you?

You can talk with other users over the network by establishing a two-way exchange of messages. Either you or the other person starts the dialogue by sending a message out and waiting for a response.

You can turn your messages off so other users can't talk to you or interrupt you. This has the same effect as taking your phone off the hook. To turn your messages off, type **mesg n**.

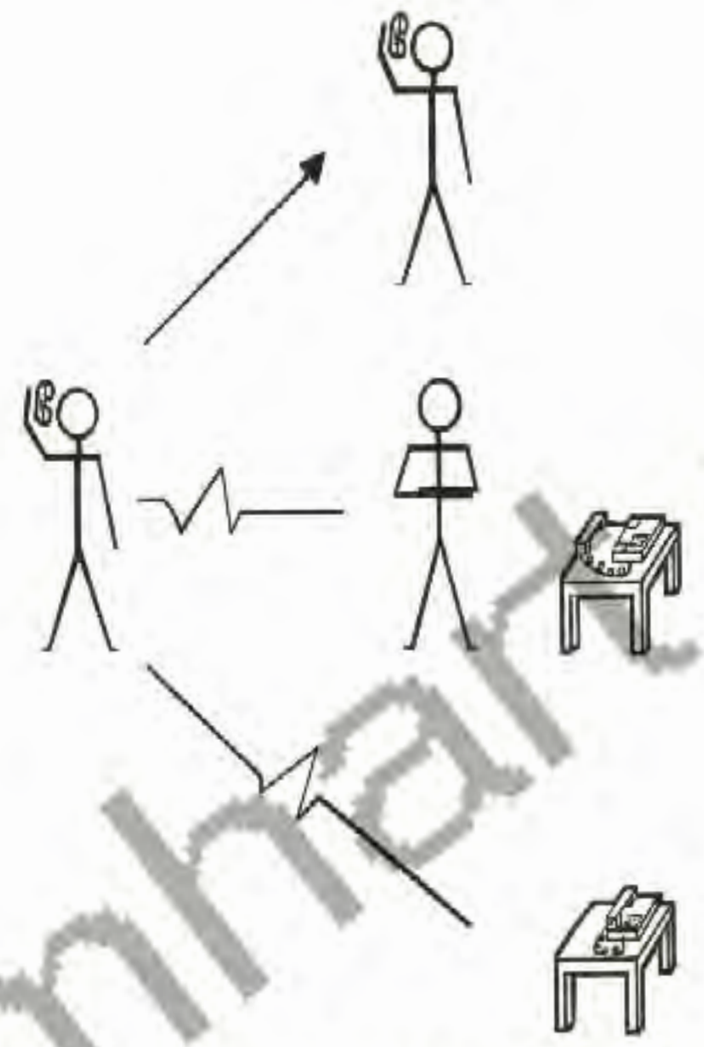
Check that your messages are off by typing **mesg** alone.

To turn your messages back on, type **mesg y**.

```
$ mesg n
$ mesg
mesg is n

$  mesg y
$  mesg
mesg is y
```

Talk with another user over the network

The **talk** command lets you establish a dialogue with another user on the network (or on your personal system). After you establish the dialogue, you can communicate back and forth by typing at your computer.



Talk to users who are logged in and answering

Start talk by typing **talk *username***. If the user is on another system, you must append the *systemname* to the *username*.

```
$ talk username@systemname
```

Username is the user's login name and *systemname* is the name of the user's system. After you type this command, the screen clears and the **talk** screen appears.

```
[Waiting for your party to respond]

------------------------------------------------------------------------
```

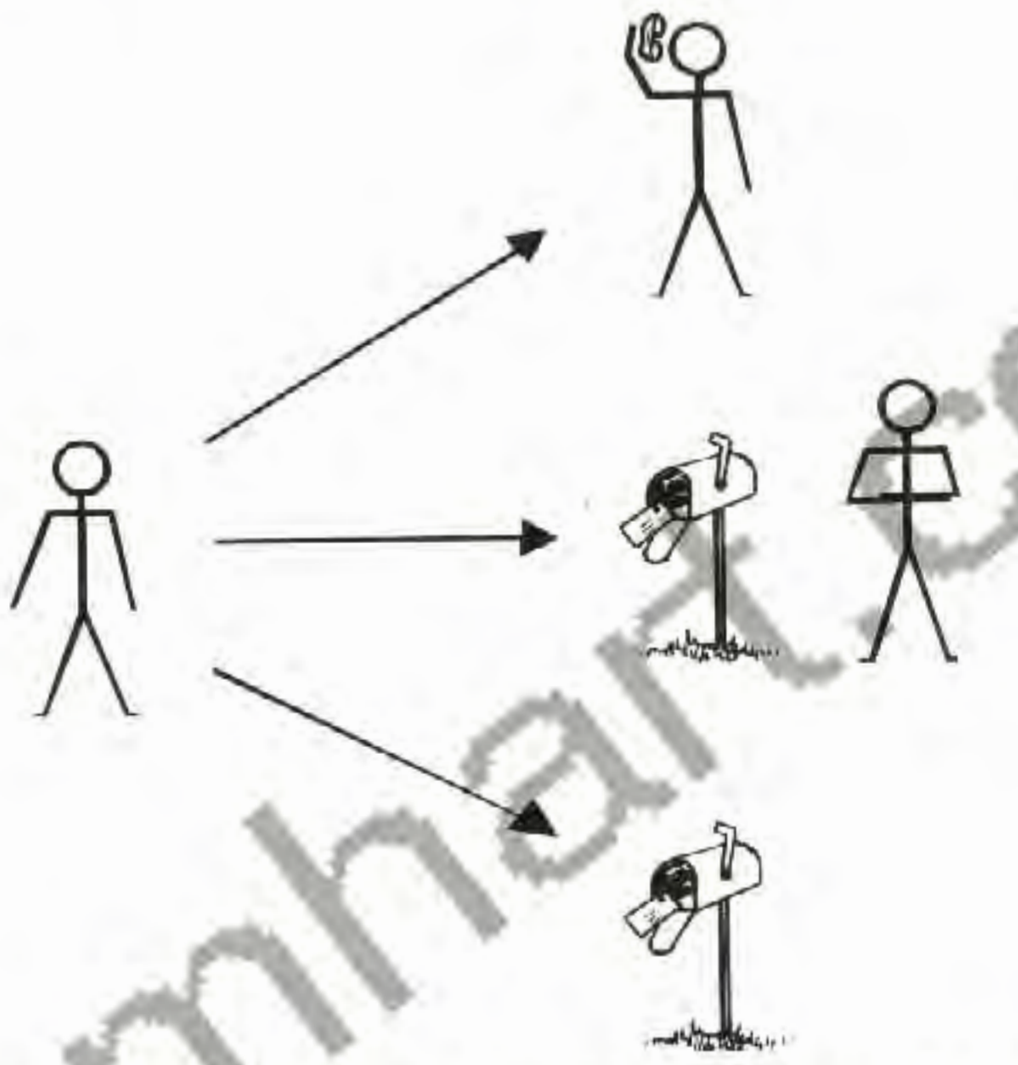
Type on the top; read the bottom

The **talk** screen is divided into two parts. The top half is where you type and the bottom half is where you see what the other person types. Note the first message at the top of your screen. You can't start communicating until the other person responds to you.

Once you establish communication with the other user, you can type messages back and forth. When you want to end the **talk** session, press CTRL-C.

Send mail

Mail lets you send messages without disturbing the recipient. In fact, the person to whom you send **mail** doesn't even have to be logged into the system. You can send **mail** to someone on your system or another system.



Send mail regardless of whether user is logged in and answering

talk is an active communication system, while **mail** is a totally passive communication system. When you send **mail** to someone, it goes into that person's mailbox. The next time that person logs in, his shell tells him that mail is waiting. If he never reads his mail, he will never see your message.

Amiga UNIX includes a public domain mail program called **elm**. See *Using electronic mail* in this book for information about sending and reading mail. You can also use the standard mail program, described briefly in the same chapter.

Check to see if other systems are active

Before you log into a remote system, you might want to check and see whether it is active. You use the **ping** command to do this. **ping** works by sending a request over the network and telling you if the remote system answers. To use **ping**, you need the name of the other system.

```
$  ping systemname
systemname is alive
```

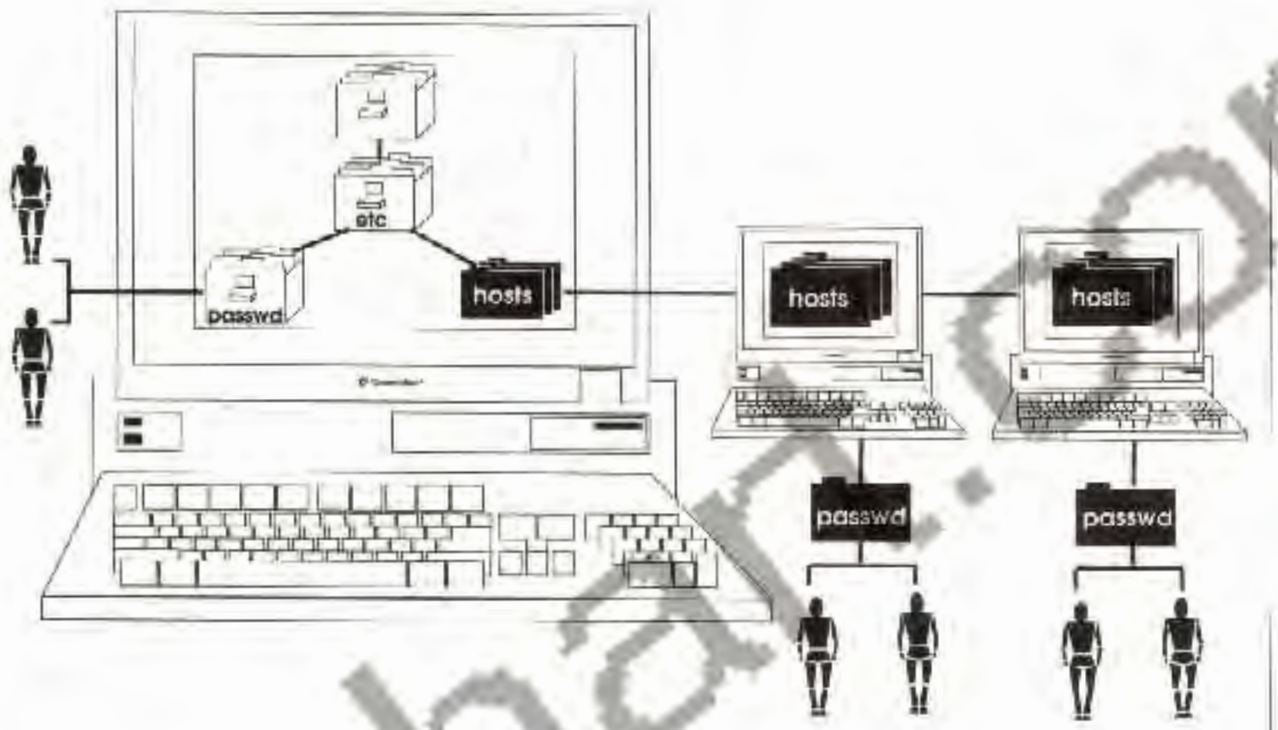
Use finger to check on remote users

You can also check to see what users are doing on a remote system before you log in to it. Use the **finger** command to get information about users on a remote system.

```
$  finger @systemname
NAME     GROUP  FULLNAME   WHAT   IDLE    TTY   LOCATION
joe      other  Joe Jones  ksh    3:01    pts/5 amiga1
sue      other  sue Jones  csh    4:02    pts/6 amiga2
```

Access a remote computer

To use a UNIX operating system, you must first log in to it. This goes for your personal system and any other system over the network to which you have access. Even though you're logged into your own system, you still have to identify yourself and be accepted by any other system you want to use.



passwd file lists users; hosts file lists systems

Use the remote login command

You don't always have to type your username and password to log in to a remote system. You could use a command called **rlogin** (remote login) to automate the login procedure.

How does rlogin work?

You must have a user account on the other system. Both systems also need a hosts file in their /etc directories that list your system name and number. Once the other system knows who you are (through /etc/passwd and /etc/hosts) you can type **rlogin *systemname*** to login into the remote system. The remote system still asks for your password, but it knows your username.

Create a .rhosts file

To further automate the rlogin procedure, create a .rhosts file in your home directory on the remote system.

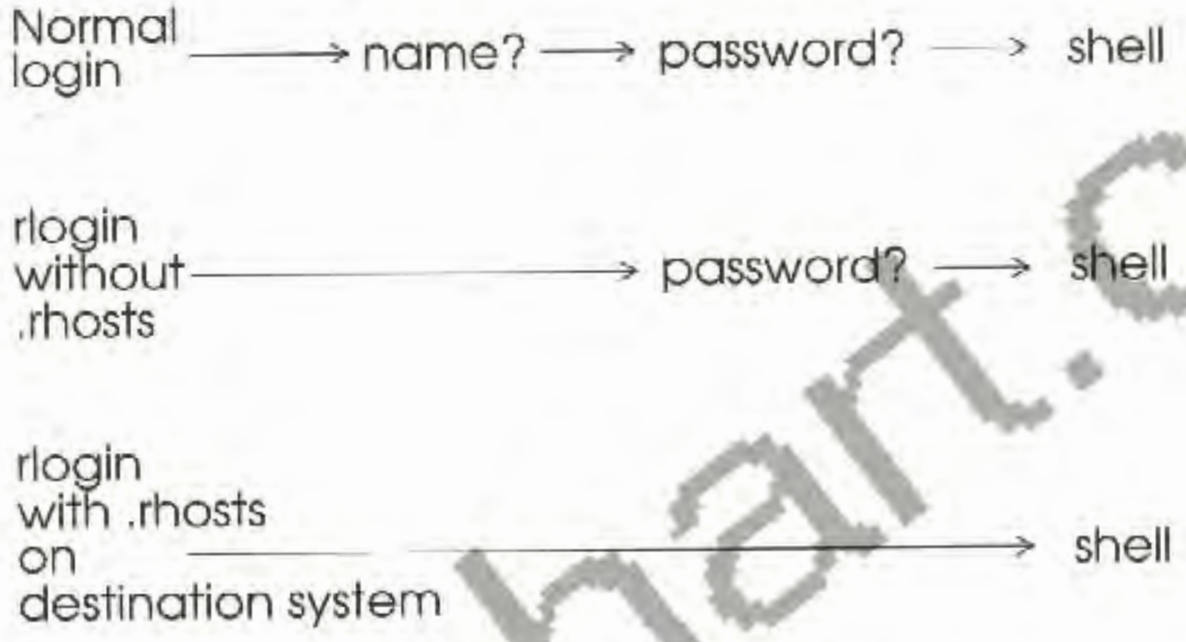
Use an editor (such as vi) to create the file. Put your .rhosts file in your home directory on any system you login to or want to exchange files with. If you want people from that system to access your system, put their .rhosts on your system. The .rhosts on the destination system should identify the system from which you type the **rlogin** command.

```
amiga     joe
system3   robin
```

.rhosts on a remote system

The .rhosts lets you login by typing the system name.

```
$  rlogin systemname
```



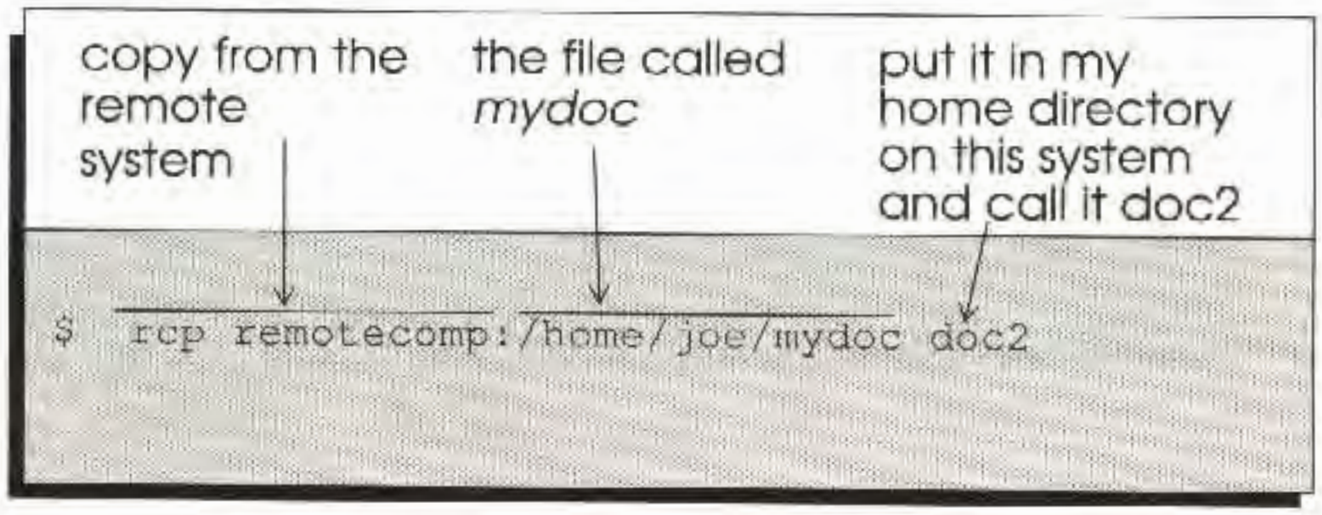
login shortcut through rlogin

Copy files over the network

The third chapter of this book discusses copying files locally using **cp**. You can do the same thing globally using **rcp**. **rcp** stands for remote copy and works much the same way as **cp**.

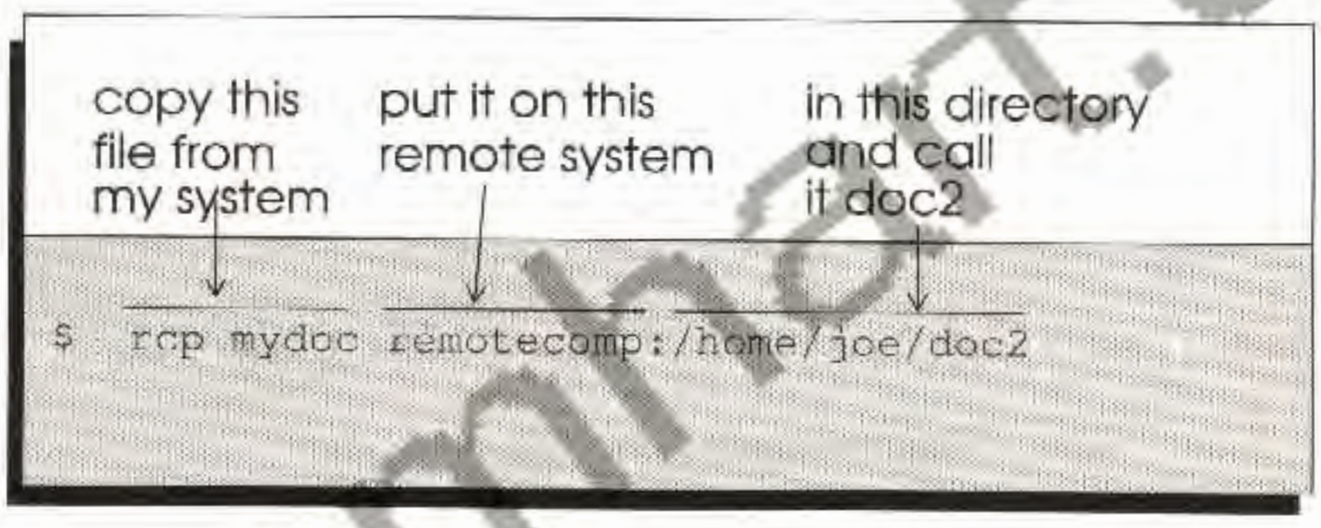
You can copy files to other systems

To copy files over the network, type **rcp** ***systemname:filename filename***. The examples on the following page illustrate some uses for the **rcp** command.



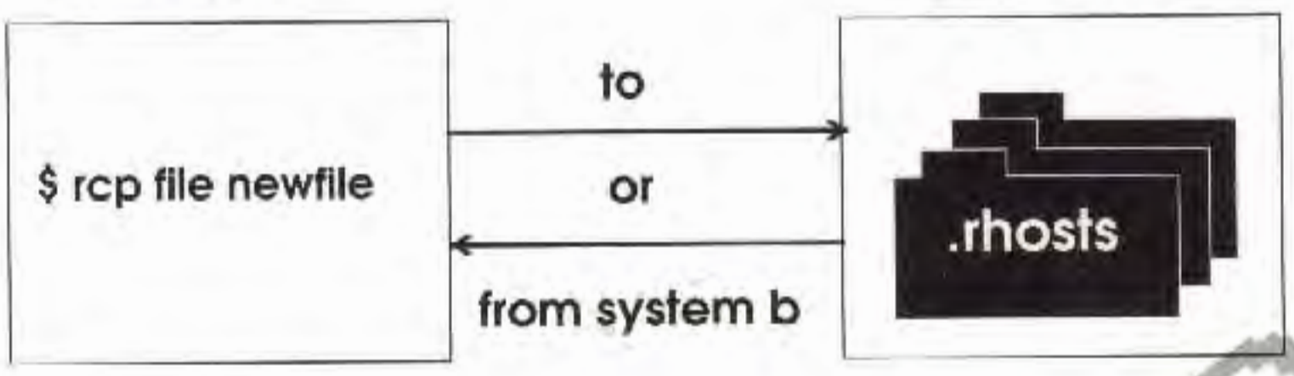
Copy files to another system

You can also **rcp** files from your system to another system.



rcp needs a .rhosts

In order to **rcp** files, you must have a .rhosts file in your home directory on the remote system.



Put .rhosts on the other system to recognize you

Setting up a network

Join a network vs creating one

You may already have a network to which you want to connect your Amiga. If this is the case, you need to get some information from the network administrator. Your computer's name and address must not conflict with any other systems on the network.

You can create a network if you do not already have one. The steps for creating a network are basically the same as connecting to an existing one, except that you are the network administrator and you decide what addresses to use.

What makes up a network?

These are the things you need to join or create a network:

- a physical connection to the network
- a unique system name
- a unique system number (address)
- a hosts file (your "directory" to the other computers on the network)
- a user account on any remote system(s) you want to use
- a .rhosts file on remote systems to allow remote file copies and to simplify remote logins

These requirements are described in the next few pages.

The physical connection

The physical connection to your network depends on the type of network you are using. For an Ethernet or TCP/IP network, you have to install an expansion card inside your system, run a special cable to this card, and connect it to the network.

You could also have a serial network where you connect a special cable to the Amiga's built in serial port. Using the built-in serial port to connect computers is probably the easiest and least expensive way of networking. Serial networks have fewer features than other types of networks, which limits their usefulness to being just another login screen.

Dumb terminals connected to your system (serial)

Systems networked together

Different ways of working with other users

The system address and hosts file

Each system on a network must have a unique system number and name. Think of a telephone directory. When someone wants to call you they find your name in the phone book, read your phone number, then dial it. Your hosts file (located in /etc/hosts) serves this same purpose. You can't contact a computer that is

not listed in your hosts file. Also, other people can't contact you if your address isn't listed in their hosts file.

See your system administrator for a valid address

If you are connecting to an existing network, be sure to ask your system administrator for your number. Don't just make one up. You should also check to see that no one else is using your system name. Your system administrator should give you a valid address and put your name and address in the other network hosts file. He or she should also give you a copy of the hosts file. If you are creating your own network, see *Editing system files* later in this book for more information about the hosts file.

You can talk to a remote computer, as long as you both have a hosts file. This hosts file must contain both your system information and the remote system's information.

In order to log into a remote system, you must have a user account on that system. You may be connected to hundreds of systems, based on your hosts file and your network wiring, but each system's owner decides whether to let you log into that system.

You can let other people log in to your computer by setting up user accounts for them. For information on setting up a user account, see the chapter called *Maintaining your system*.

Administering a network

Advanced networking topics

Networking is much more complicated than we can cover in this chapter. You should, however, now have a simple picture of how to work over a network, both locally and globally. There are many other networking tasks that you could spend months learning, and numerous commands that you can perform remotely.

Network administation tasks

If your computer is connected to a large network, you might need to perform periodic network administration tasks. These tasks include:

- setting up hardware for new users
- expanding the size of your network
- adding user accounts to your and all other systems
- adding new systems to your hosts file
- making specific files and directories available to other users
- mounting file systems from other systems
- sharing printers

The best way to learn more is to read an advanced networking guide and experiment with your network.

Getting more information

Networking with UNIX is fairly generic, which is why you can connect so many different systems together. Many of the world's most popular networking schemes are included in Release 4 UNIX and other UNIX enhancements. Because UNIX networking is so generic, there are many good books on the subject. If you want to dive deeper into the world of networking, check your local bookstore for UNIX networking books.

Basic skills are enough

After reading this chapter, you should be fully equipped to communicate with other users and systems. The list of topics above are useful, and at times important, but they are not critical to basic networking.

Troubleshooting

Problem	Solution
Can't rlogin to a remote system	You're not in the .rhosts (or you don't have a .rhosts) file in your home directory on the remote system. Check the physical connections. You don't have a user account on the remote system. You typed your login name or password incorrectly. Your /etc/hosts file doesn't contain the name of the system you are trying to login to. Use the **ping** command to see if the other system is "alive". It might be turned off or disconnected from the network.
Can't talk to another user	The other user has his or her messages turned off (**mesg** n). The other user is not logged in. The other user doesn't want to respond.
Can't broadcast a message to other users	The other users have their messages turned off (**mesg** n). You are not logged in as root.

Problem	Solution
mail won't work	Make sure you specified the right system and user name. Read your mail; mail sends you a message explaining why it can't deliver your message.
Can't copy a file to or from another system	You don't have a user account on the remote system. You don't have .rhosts in your home directory on the remote system. Your .rhosts file on the remote system doesn't recognize you or your current system.

Special features of Amiga UNIX

Amiga UNIX commands

color	set or show screen color
fdfmt	format a floppy disk
getscr	define and create a virtual screen
passwdall	set/erase all system account passwords
rdb	set/show hard disk partitions
sioc	set/display keymap, font, and screen size

Amiga UNIX directories

Special Amiga UNIX programs and utilities are stored in the following directories.

/usr/amiga/bin
/usr/amiga/etc
/usr/amiga/lib/kmap
/usr/amiga/lib/font
/usr/public/bin

Public domain programs

Amiga UNIX includes several public domain alternatives to standard System V Release 4 programs. The System V Release 4 programs still exist and work properly.

Unix standard	Amiga UNIX alternate
mail	elm
more	less
finger	Finger
vi	emacs
cc	gcc

Special features of Amiga UNIX

Why should you read this chapter?

Your Amiga UNIX system is a complete version of AT&T's UNIX System V Release 4.0 (Release 4) and provides all the features of that operating system. However, Amiga UNIX is more than just Release 4; it combines elements from several sources:

- Amiga high resolution graphics
- Amiga enhancements, including virtual screens, device drivers, and system-specific hardware functions
- AT&T UNIX, which includes Berkeley and Xenix commands
- public domain utilities

Read this chapter to learn how Amiga UNIX differs from other versions of Release 4 UNIX. You can also read man pages for any of the special Amiga UNIX programs. The regular **man** command works for most Amiga enhancements.

Where are the Amiga UNIX programs?

Amiga UNIX programs and utilities that are not available on any other system are in the following directories:

- /usr/amiga/bin
- /usr/amiga/etc
- /usr/amiga/lib/kmap
- /usr/amiga/lib/font
- /usr/public/bin

What features are unique to Amiga UNIX?

Amiga UNIX enhancements cover four distinct areas:

- screen management
- online guided interface to user and administrator tasks
- public domain utility programs
- miscellaneous utilities and system calls unique to the Amiga port of the UNIX operating system

Screens and display options

Screen management programs define and create virtual screens, allow access to the Amiga console and custom RAM, and control various display elements of Amiga screens, including color, font type, and character size.

Public domain programs

We added several public domain programs, and even documented them as primary options, because they are well known and, in some cases, more useful than the comparable Release 4 programs. The Release 4 programs still exist, work properly, and are located in the right place; we have simply added alternatives, as shown in the following table.

UNIX standard	Amiga UNIX alternate
mail	elm
more	less
finger	Finger
vi	emacs
cc	gcc

Other public domain programs provide functions not included in standard UNIX Release 4. These programs are in the directory /usr/public/bin.

Unique Amiga UNIX functions

Some Amiga UNIX features are unique only because of the Amiga hardware. Functions such as formatting floppy disks and hard disk partitions, connecting a parallel printer, and setting a keyboard translation map exist in some form on many systems; the specific variants for Amiga UNIX are documented in this chapter. Again, any unique features are in addition to UNIX Release 4; no AT&T, Berkeley, or XENIX functions have been removed or replaced.

Minor changes have no effect

Some programs were modified as part of the porting process. These minor modifications are transparent to users, programmers, and administrators. Most importantly, they do not interfere with the pure binary compatibility that defines a successful Release 4 product. These modifications are intrinsic to the porting process and have no effect other than to make UNIX work properly on an Amiga.

What are the screen functions?

Screen functions are stored in the /usr/amiga/bin, /usr/amiga/lib/kmap, and /usr/amiga/lib/fonts directories. Screen functions cover the following areas:

- defining virtual screens
- using virtual screens
- listing virtual screens
- changing the background and foreground colors
- setting the font size and type
- mapping one of a variety of international keyboards to the current character set
- displaying high resolution graphics on an Amiga or Moniterm monitor

Are screen commands necessary?

Note that the X Window System and Release 4 graphic functions are supported by Amiga UNIX without modification. Amiga UNIX provides virtual screens and customized displays as additions to standard UNIX; you don't need to use these enhancements, or any special Amiga UNIX features, to operate X or UNIX. Default display options for all screens are already defined; you can use Amiga UNIX, its virtual screens, and its graphic capabilities without ever learning any of the special Amiga commands.

Virtual screens

Check the size of your screen

Based on the resolution of your screen (either what you specified in /etc/inittab or what your monitor allows) and your font size, Amiga UNIX calculates the number of fixed space columns and rows that can fit on the screen. Use **sioc winsize** to see how many lines and columns are on your screen.

```
$ sioc winsize
LINES=25 COLUMNS=80
PLINES=200 PCOLUMNS=640
```

Amiga A2024 and Moniterm monitors

You can use a special Amiga high resolution display setting if you have a Commodore A2024 monitor or a Moniterm monitor. These monitors have a graphics board that enhances the Amiga's graphic output for higher resolution.

Define high resolution for your system

During installation, you tell your system if you have one of these monitor types. You can subsequently change this setting with an **sioc** command that turns high resolution on or off. **sioc setdisplaytype +4** turns the high resolution on for the current session; **sioc setdisplaytype -4** turns it off. These commands are not permanent; the system returns to its installed state when you reboot. You can change the installed state by linking the file /usr/amiga/bin/A2024 to either /bin/true (if you have an A2024 or Moniterm) or /bin/false.

```
$ ln /bin/true /usr/amiga/bin/A2024
```

Create a high resolution virtual screen

You set the high resolution mode for a particular screen in /etc/inittab by using the highest x and y settings (x=1024, y=800 or 1024). As with any inittab line, it will not be activated until you type **init q** to force a read of the inittab, and log out and log back in on that function key.

Create a high resolution X screen

You can also create a high resolution X Window System screen by using the **Xamix** or **olinit** commands. These commands test /usr/amiga/bin/A2024 to determine if high resolution is allowed; if it is, they create the X screen at this resolution.

Set overscan parameters

Even without this high resolution mode, you can increase the size of your screen by using overscan. Many monitors do not support overscan, which involves writing up to 64 extra pixels to both sides and the top and bottom of the screen. If your monitor has overscan, and you want these extra pixels, modify the file /etc/rc2.d/S70sioc. Near the end of this short startup file is a commented out line that sets the overscan for each of the four screen edges.

Mapping the keyboard to a character set

Your system uses a keyboard map

Your system is installed with a keyboard map that tells Amiga UNIX what character to use when a specific key is pressed. You can set both a system default and a specific override for your current session.

The standard keyboard maps provided with Amiga UNIX are listed below.

Key map	Country
cdn	French Canadian
ch1, ch2	Swiss
d	German
dk	Danish
e	Spanish
f	French
gb	British
i	Italian
is	Icclandic
n	Norwegian
s	Swedish/Finnish
usa0, usa1	United States options
usa2	United States Dvorak keyboard
usa	United States (default keyboard)

Set the keyboard map

Use **sioc** to change either the current keyboard map or the default map. **sioc setkmap *filename*** changes the current map; **sioc setkmap** resets the default map; and **sioc setdefkmap *filename*** changes the system default keymap.

```
$ sioc setdefkmap /usr/amiga/lib/kmap/usa1
```

Amiga UNIX utilities

What are the Amiga-specific programs?

The directories /usr/amiga/bin and /usr/amiga/etc contain programs that are unique to Amiga UNIX, most of which are designed specifically for Amiga hardware. The most important programs in these directories are listed below.

Program	What does it do?
A2024	linked to true or false to allow or disable high resolution
color	sets or shows the screen color
fdfmt	formats a floppy disk
getscr	defines and creates a virtual screen
passwdall	sets or erases password for all system accounts
rdb	sets and shows partitions on a hard disk
sioc	sets and displays current and default key map, font, and screen size

A2024 and **sioc** are described earlier in this chapter. **color** and **getscr** are described in the *Getting started* chapter. **rdb** is described in the *Maintaining your system* chapter.

System passwords

passwdall is run automatically, as part of the installation procedure, to assign a password to the system accounts or to remove an existing password from those accounts (**passwdall -d**).

Format a floppy disk

fdfmt is a single-line shell script that executes a **dd** command with all the options to format a floppy disk; the syntax to format a floppy disk in the first floppy drive is **fdfmt > /dev/rdsk/fd0f**.

Editing system files

About system files

Some system files are customizable files that Amiga UNIX needs to function. Use them to customize your environment, add users to your system, and name your system.

Only root can edit system files.

System files are usually maintained by system administrators.

Edit system files with an editor such as vi.

Contents of the system files

/etc/passwd	Identifies users who are allowed to log in to the system.
/etc/group	Identifies users who are logically grouped together and who share the same access privileges.
/etc/profile	Contains information that the user's shell reads each time a user logs in. Sets up that user's environment for the login session.
/etc/inittab	Contains settings for init levels, virtual screens, and serial devices.
/etc/vfstab	Contains mount information about hard disks and file systems.
/etc/hosts	Contains information about other computers on your network. "Telephone directory" to the network.
/etc/nodename	Contains your system name.

Editing system files

What are system files?

UNIX system files are special files that only root can modify. The system files affect the way UNIX behaves. You use them to customize your environment, add users and systems to your system, and name your system. If you are a user, and not a system administrator, you can skip this chapter. If you are the administrator of your own system, you need to understand the system files so you can look at and occasionally change them.

Here's a list of the system files you should know about:

- /etc/passwd
- /etc/group
- /etc/profile
- /etc/inittab
- /etc/vfstab
- /etc/hosts
- /etc/nodename

How do you edit a system file?

You edit most of these system files by opening the file with an editor (such as **vi**) and adding a line to it. This section shows you what information you can add or change in each file and what this information means.

You must log in as root in order to edit system files. Anybody can look at most system files, but only root can change them.

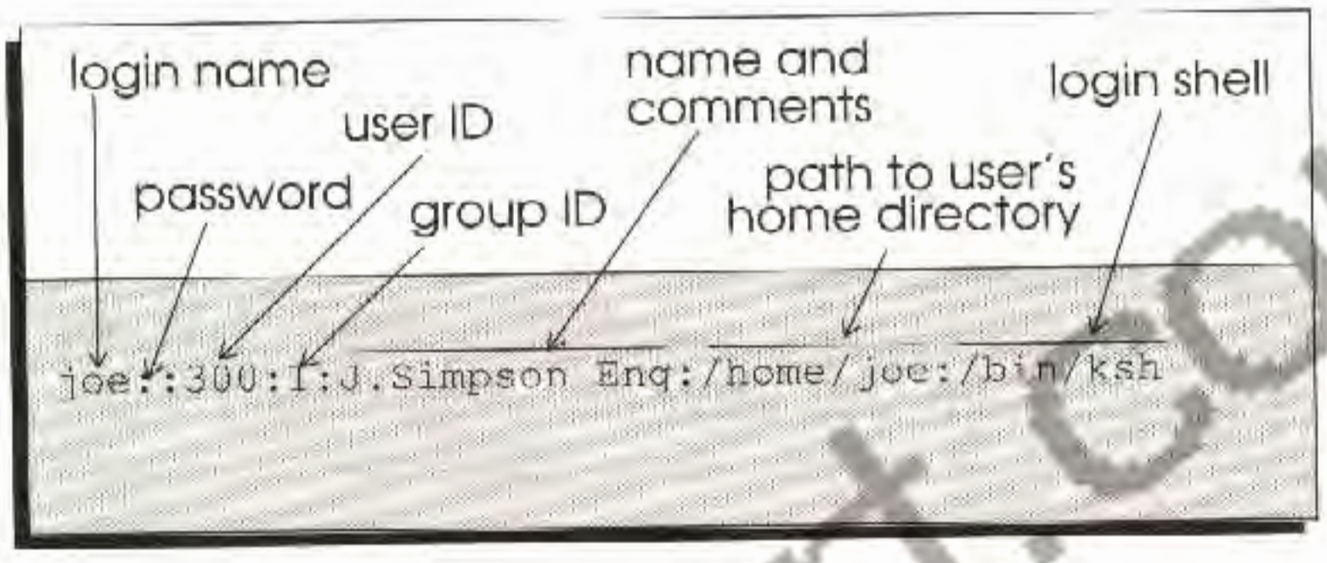
NOTE: Use extreme caution when changing system files. If you damage or lose a system file, you may have to reinstall Amiga UNIX.

Adding users to /etc/passwd

Add a user to /etc/passwd

The passwd file (pronounced password) is where you identify users who are allowed to log into your system. To add a user, you add a line for that user in the /etc/passwd file.

The following example shows a typical line from the /etc/passwd file.



Field	What is it for?
login name	lower case, eight characters or less
password	leave it blank; you add it later
user ID	greater than 100, less than 60001, no duplicates in the file
group ID	any number from /etc/group
comment	full name, phone number, comments
home	home directory (/home/*username*)
login shell	shell that starts on login

Passwords are in /etc/shadow

There's another file, /etc/shadow, that contains the passwords for all users. Only root can read /etc/shadow, and you must run a program called **pwconv** to update this file each time you edit /etc/passwd.

Adding groups to /etc/group

Why use groupnames?

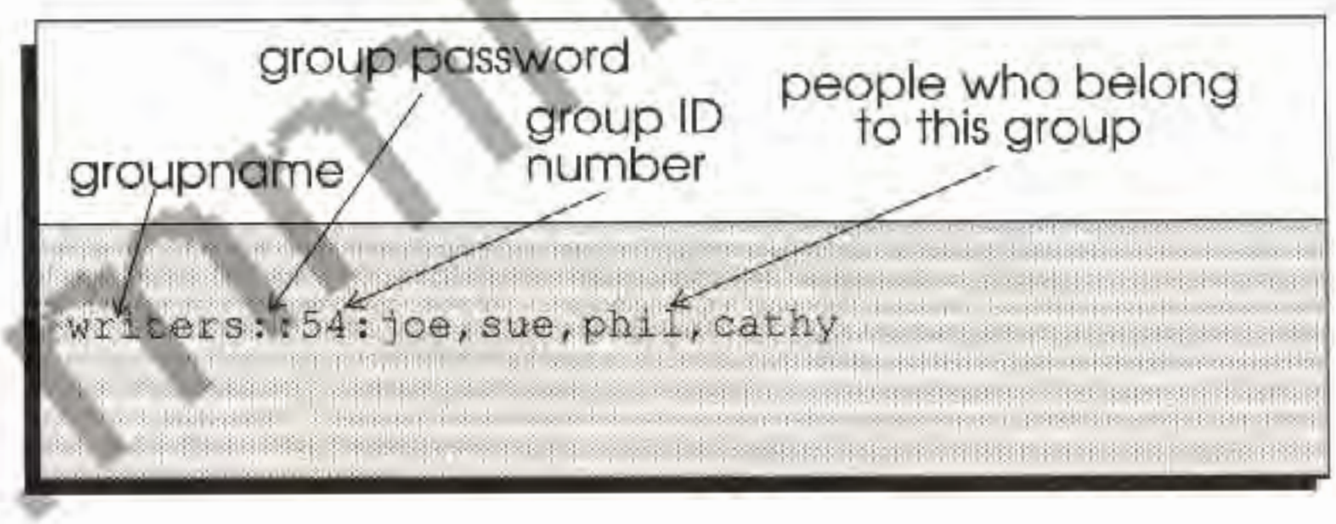
Each user belongs to a group. Group numbers are as unique to groups as user numbers are to users. Your privileges in the UNIX system are based on your user ID and your group ID.

Some users are logically grouped together; all the programmers, all the writers, or all the salespeople. Each member of a group has the same group privileges, which may or may not be the same as those of other groups.

A primary advantage to setting up groups is to share files and common directories. You can make it so only people in your group can share your files. If someone from a different group tries to look at your files, UNIX denies them access.

You can add as many groups as you need, any time you need. You do this by adding a line to /etc/group.

What's in /etc/group?



Field	What is it for?
groupname	meaningful name for the group
password	leave this field empty
group ID	any group number that is not in use
usernames	login names of people who belong to this group

Defining startup actions in /etc/profile

What is /etc/profile?

The file /etc/profile contains information that the shell reads each time you or someone else logs in. The information gives the shell a "profile" of how to start a standard login session.

NOTE: /etc/profile only affects sh, ksh, and rsh. It has no affect on csh.

/etc/profile also automates some common tasks, such as displaying the message of the day. Amiga UNIX uses /etc/profile to configure your system's default environment; your own .profile then modifies this standard environment to suit your personal needs.

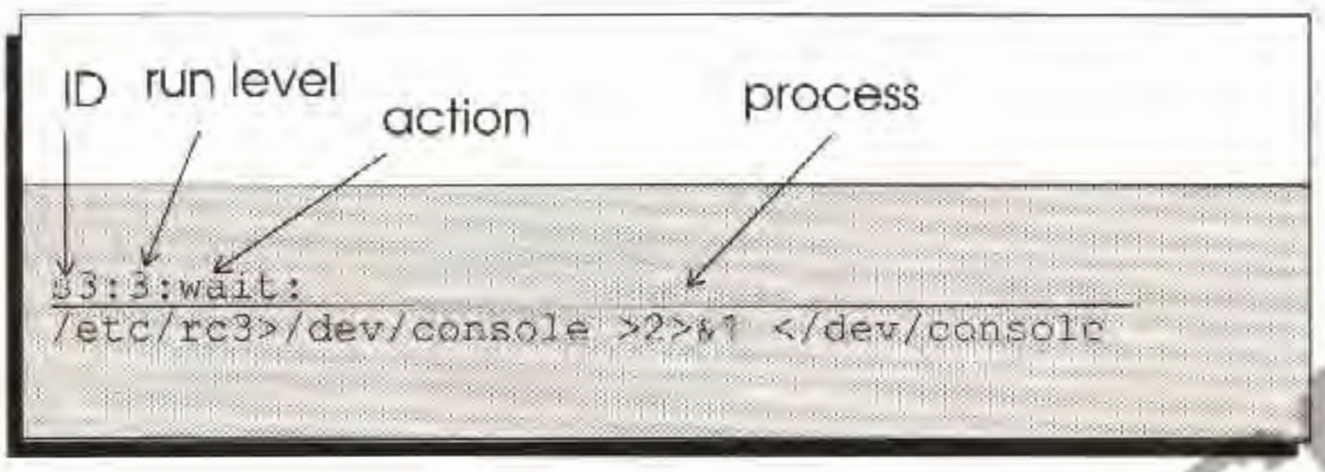
Here are some of the things the default /etc/ profile does:

- set the search path
- set the timezone
- set the terminal type and characteristics
- display the message of the day
- test whether a user has mail
- set default permissions for new files

Defining devices in /etc/inittab

Define devices and system processes in /etc/inittab

/etc/inittab contains information about init levels, virtual screens, and serial devices. Each line does something different, but they all share a common format.



Field	What is it for?
ID	name of this line
run level	init level in which action takes place
action	tells init how to handle process
process	command to execute at this init level

Init levels in /etc/inittab

UNIX is always running in one of 7 modes (0-6 or s). **s** puts it in single user mode, **2** puts it in multi-user mode, **0** shuts it down completely, and 6 executes a reboot to multi-user mode. These modes are defined in /etc/inittab and are called init levels. There are also three undefined init levels (a, b, and c) that you can change to create special ad hoc init processes.

You change init levels by running the **init** command.

For example, to bring your system to single user mode, login as root on F1 and use the init option for single-user mode.

```
#  init s
```

When you run the **init** command, it reads /etc/inittab and executes the processes for the init level you specified.

Virtual screens and /etc/inittab

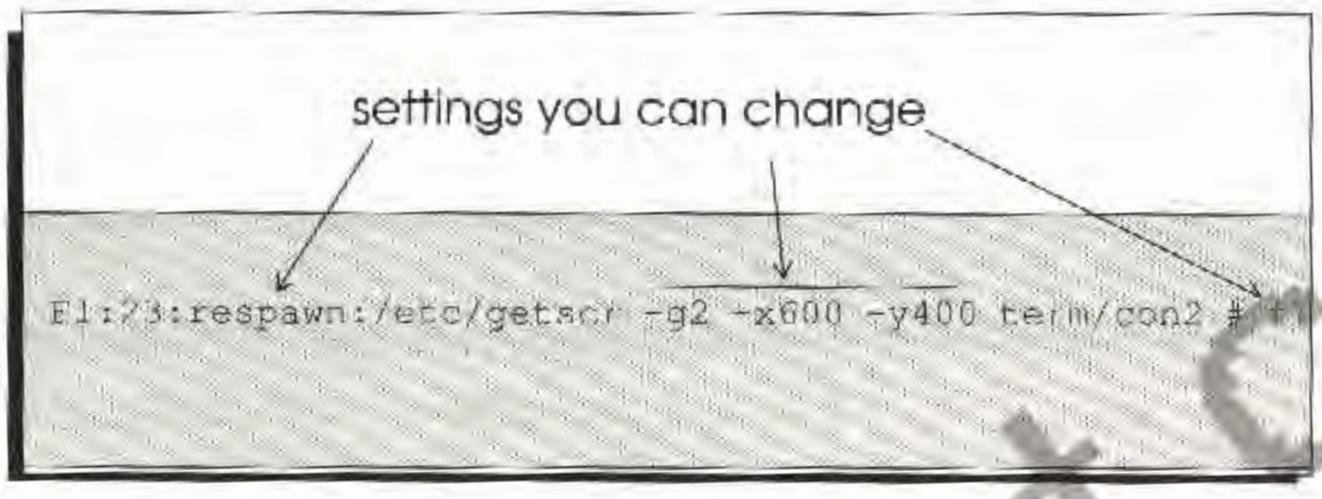
Virtual screens are defined in /etc/inittab, in different lines for each function key.

```
con:23:respawn:/etc/getty console console # system console
screen
F2:23:respawn:/etc/getscr -g2 -x640 -y200 term/con2  con  #F2
F3:23:respawn:/etc/getscr -g3 -x704 -y464 term/con3  con  #F3
F4:23:respawn:/etc/getscr -g4 -x704 -y464 term/con4  con  #F4
                .
                .
.F10:23:off:/etc/getscr -g10 -x1024 -y1024 term/con10  #F10
```

Which virtual screen settings do you change?

You customize your virtual screens by changing these lines. You change the settings in these lines to change the characters and colors on your monitor.

The following example shows which parts of a virtual screen inittab line you can change.



Virtual screen options

Option	What does it do?
respawn/off	Turn screen on or off
-g *number*	function key (2 through 10)
-x *number*	horizontal pixels (usually 320 or 640, or 1024 if you have Amiga high-resolution hardware)
-y *number*	vertical pixels (usually 200 or 400, or 800 if you have Amiga high-resolution hardware)
-0 *xxx*	background color, from 000 (black) to fff (white)
-1 *xxx*	foreground color (000 to fff)
-f *filename*	font type and size (from the /usr/amiga/lib/fonts directory)
term/*conx*	name of screen (con2 through con10)

Serial devices and /etc/inittab

You can set the options for each of the nine default login screens (2 through 10, since function key 1 always returns to the system console). If you turn a screen on (respawn) or off, you must type **init q** for init to read the /etc/inittab file and notice the change.

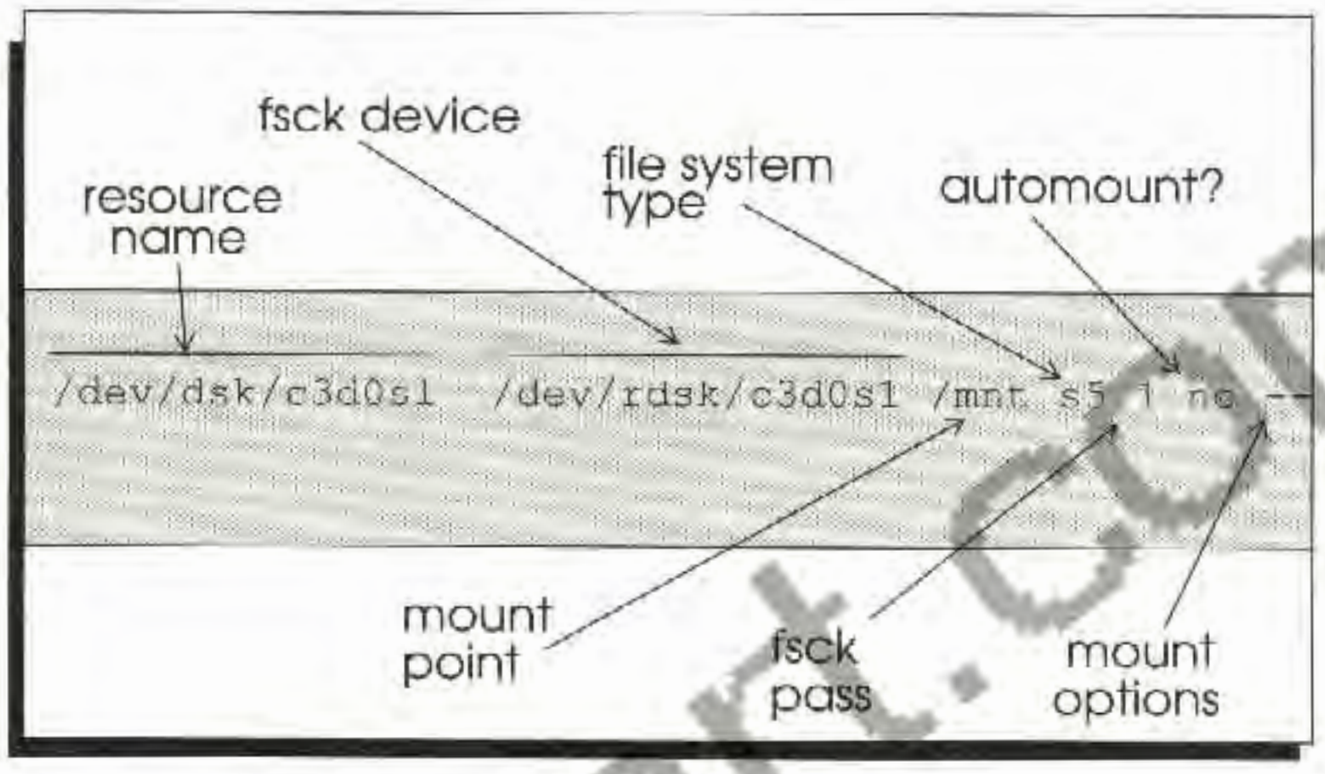
You configure the default serial port (ser0) and any others you add (such as those on the A2232 card) in /etc/inittab. You can specify the serial device name and the baud rate. The following screen shows how you might configure the main serial port for a dumb terminal.

```
ser0:23:respawn:/etc/getty term/scr 9600 #ser port
```

Define disks and file systems in /etc/vfstab

Edit vfstab

The /etc/vfstab file contains information about hard disks and file systems. You edit vfstab when you add a hard disk. UNIX then reads vfstab to automatically mount the hard disk when the sytem boots.



Field	What does it do?
resource name	block device name
fsck device	character device name
mount point	directory in root file system where you want this disk mounted
fs type	type of file system (s5 is SystemV)
fsck pass	number for check, - for no check
automount?	y to mount automatically at boot time
mount options	special options to execute when the system mounts a file system

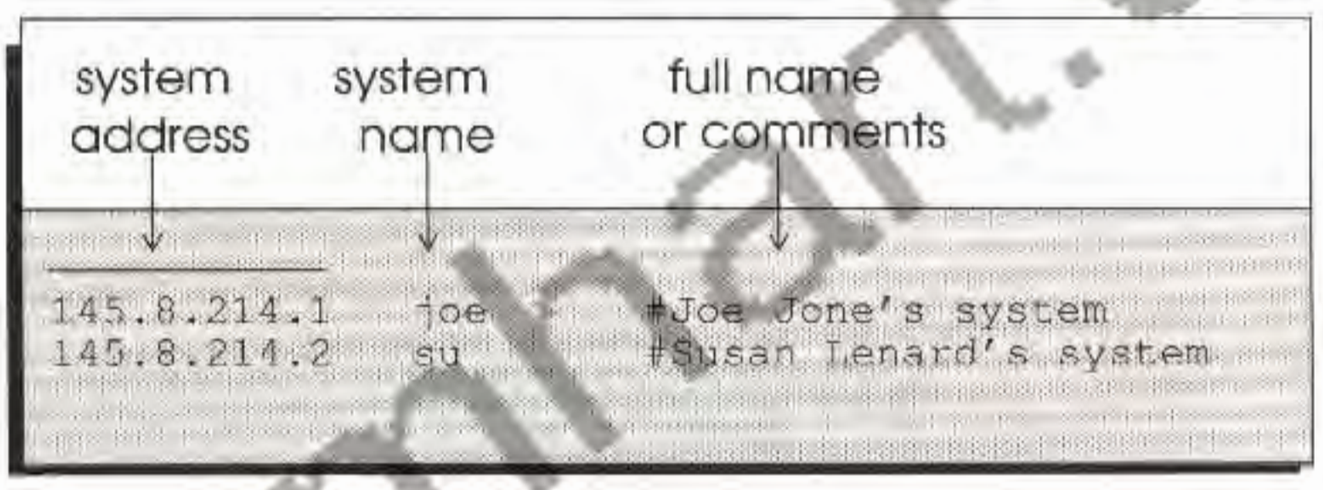
Network system names in /etc/hosts

Add network system names to /etc/hosts

The /etc/hosts file contains information about other computers on your network. This hosts file is like a telephone directory; it lists a unique name and address for every computer in your "world". Each computer connected to the network must also have a copy of /etc/hosts, with all the computers it recognizes listed in this file. For convenience, most network administrators put the same hosts file on all the major computers and route the lesser computers through these points.

During installation, you may have set up a two computer network. If so, the two systems you added appear in the hosts file.

What do you put in /etc/hosts?



Field	What is it for?
system address	unique number
system name	name of your system
comments	has no effect on the hosts line; you can put a user's location or anything else you like; must start with a # sign

Naming your system in /etc/nodename

What is /etc/nodename?

The /etc/nodename file contains your system name. During the initial installation, you used the configuration script to name your system.

Use uname to set your nodename

You shouln't edit /etc/nodename directly. Use the **uname** command to change your *systemname*. This also puts your *systemname* in the other files where it is needed, including /etc/net/ticlts/hosts.

```
#  uname -S systemname
```

uname -S puts the system name into /etc/nodename and updates a couple of other important files. If you look inside /etc/nodename, you should see the new system name.

```
$  cat /etc/nodename
systemname
$
```

Don't forget to change the *systemname* in your hosts file and tell others about the change. If other users have a different name for you in their hosts file, they may not be able to communicate with or recognize you.

Maintaining your system

Creating user accounts

Log in as root	Only root can add users
Edit /etc/passwd	Add a line that identifies users who can login to this machine
Run **pwconv**	Update associated files with new /etc/passwd line
Create directory	Create home directory for new user account
Change ownership	Change directory's ownership and group from root to that of new user
passwd ***username***	Add a temporary password to user accounts

Shutting down your computer

Always run shutdown before turning your computer off. You must be root and in the root directory (/).

 # shutdown -y -g *delay* - i *initstate*

-y	Skip questions; assume answer is yes
delay	Specify the number of seconds before shutdown occurs
init state	Specify the initstate you want for your system after shutdown; 0 is power off, 6 is reboot

Maintenance terms and concepts

User accounts	Lines in /etc/passwd that identify users who can log in to this system
Backup	Copying important files and directory structures to another directory, floppy disk or tape.
Terminal	Monitor and keyboard that connects to the serial port, as an extra login screen, or dumb terminal

Maintaining your system

Why should you read this chapter

On a traditional UNIX system, one person is the system administrator and everyone else is just a user. Since Amiga UNIX is a personal workstation system, you take an active role as your own system administrator. After all, no one else is going to manage your desktop system for you.

Administering a system isn't as difficult as you might think. In fact, many "system administrators" are simply users with a special password that lets them start the sysadm menu system. Most people who use a desktop UNIX system will at some point have to perform system administration tasks.

This chapter introduces you to some techniques for keeping your Amiga UNIX system running smoothly. Unless you plan to administer a network of systems and users, this is all you need to know about system maintenance. If you are using a UNIX system that someone else is managing, you can probably skip this chapter. It covers both basic and advanced administration tasks, ranging from administrative commands you might use anytime to specific tasks you use less often.

What's in this chapter?

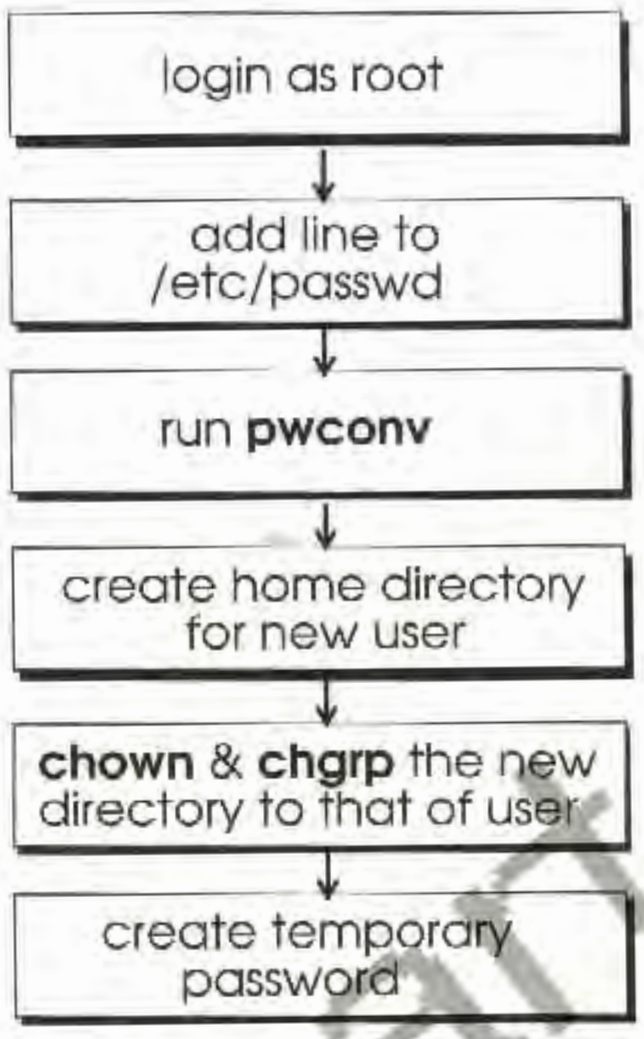
Read this chapter to learn the following tasks:

- adding and removing user accounts
- adding hard disks to your system
- adding terminals to your system
- shutting your system down safely
- backing up your files
- using **crontab** to schedule tasks

Adding and removing user accounts

Add new users

The 6 steps for setting up user accounts are:



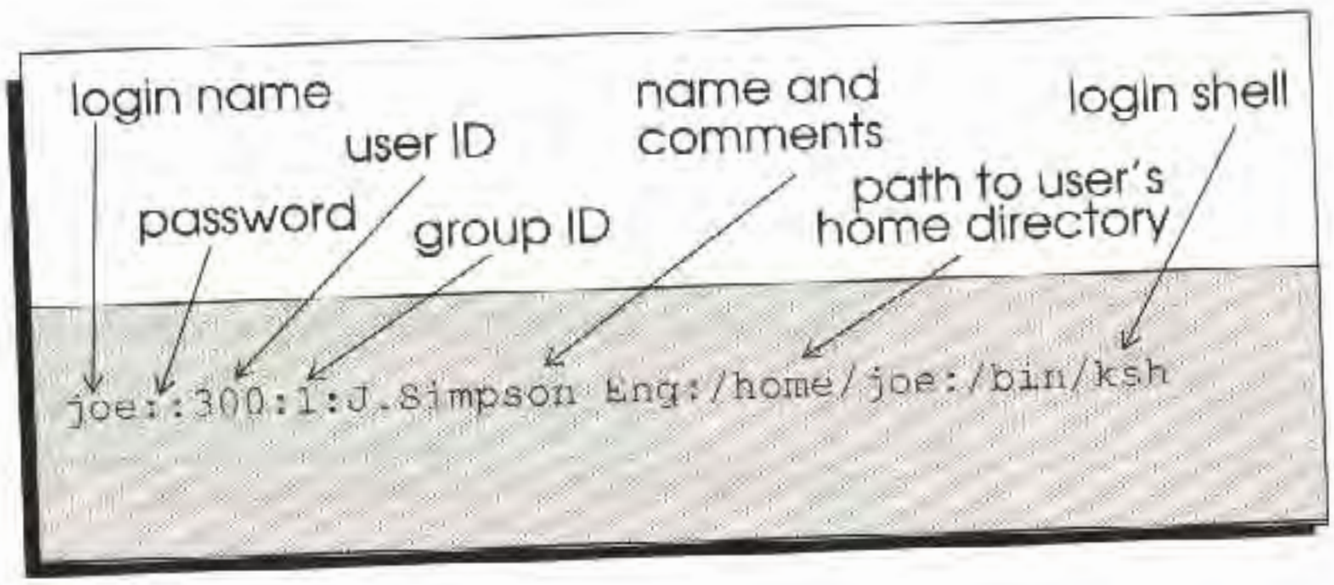
This section shows you how to add or remove a user.

The /etc/passwd file

The /etc/passwd file contains a list of everyone who has an account on your system. You control user accounts by adding, changing, or deleting lines in this file.

NOTE: The passwd file is very important to the system. If you delete or damage it, be sure you create a new one before you log out; otherwise, you'll never get back in.

Typical line from /etc/passwd



Users are defined in /etc/passwd

Field	What is it for?
login name	lower case, eight characters or less
password	leave it blank; a program adds an encrypted password later
user ID	greater than 100, less than 60001, unique
group ID	any number from /etc/group
comment	full name, phone number, comments
home	home directory, /home/*username*
login shell	shell that UNIX starts on login; /bin/sh, /bin/ksh, /bin/csh, /bin/rsh

What other files do you change?

After you enter a line for someone in your passwd file, you need to update another file, called /etc/shadow, so the two match. /etc/shadow contains the encrypted passwords for each user. If you look inside /etc/shadow, you see each user's name, an encrypted password, and a couple of other pieces of information.

Run pwconv to hide the password information

Fortunately, you don't have to worry about editing /etc/shadow, a program called **pwconv** does it for you. Run **pwconv** each time you add or remove users from /etc/passwd.

Leave the actual password slot blank in /etc/passwd. This is so that **pwconv** knows the password slot is for a new user and hasn't already been hidden. After you run **pwconv**, the following message appears:

```
pwconv:WARNING user account username has no
password
```

If you check the passwd file now, you see an x in the passwd slot. The /etc/shadow file now recognizes this user, so you can add a password.

Give new accounts a temporary password

Use the **passwd** program to add temporary passwords (such as "changeme") to the new user accounts.

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

passwd asks you to type the password twice, to confirm that you typed it correctly (since you can't see it on your screen).

Make a home directory

You specified a path to a home directory in the passwd file. You also have to create that directory. If you want to group several users together, to provide a restricted environment or make them share all their work, you can put them all in the same directory. The system does not care if each user has his own directory; it just needs to know what directory that user will be using. The system does care about access privileges, however. The directory, and all the files in it, must have the same group or other access privileges.

Use the **mkdir** command to create a home directory for new users. Usually the home directory is /home/*username*.

Change ownership of the home directory

Since you are logged in as root, this new directory belongs to root. You want to change its owner and group, so it properly belongs to the new user.

```
#  chown username directoryname
#  chgrp newgroupname directoryname
```

You now have a new user account, complete with a login name, login shell, group, password, and home directory. If you are setting up accounts for other users on your system, repeat these steps, and tell

them what temporary password you gave them. They can change their own passwords by using the **passwd** command after logging in.

Change user information

You can change a user's information any time by editing his line in /etc/passwd. Any time you change a user's information, you may have to make other changes to keep the user's world in line with the /etc/passwd file (such as move his files to a new directory or change the group on his old files).

If you change a user's name in /etc/passwd, you must run **pwconv** again. Make sure the password space in the user's line is empty before you run **pwconv**; otherwise, the program doesn't think of it as a new account.

Remove users

Follow these three steps to remove a user from your system:

- delete the line from /etc/passwd
- run **pwconv**
- use **rmdir** to remove the home directory (unless it is a shared directory)

NOTE: The **rmdir** command doesn't let you delete a directory that has files in it. You must remove all files and subdirectories first.

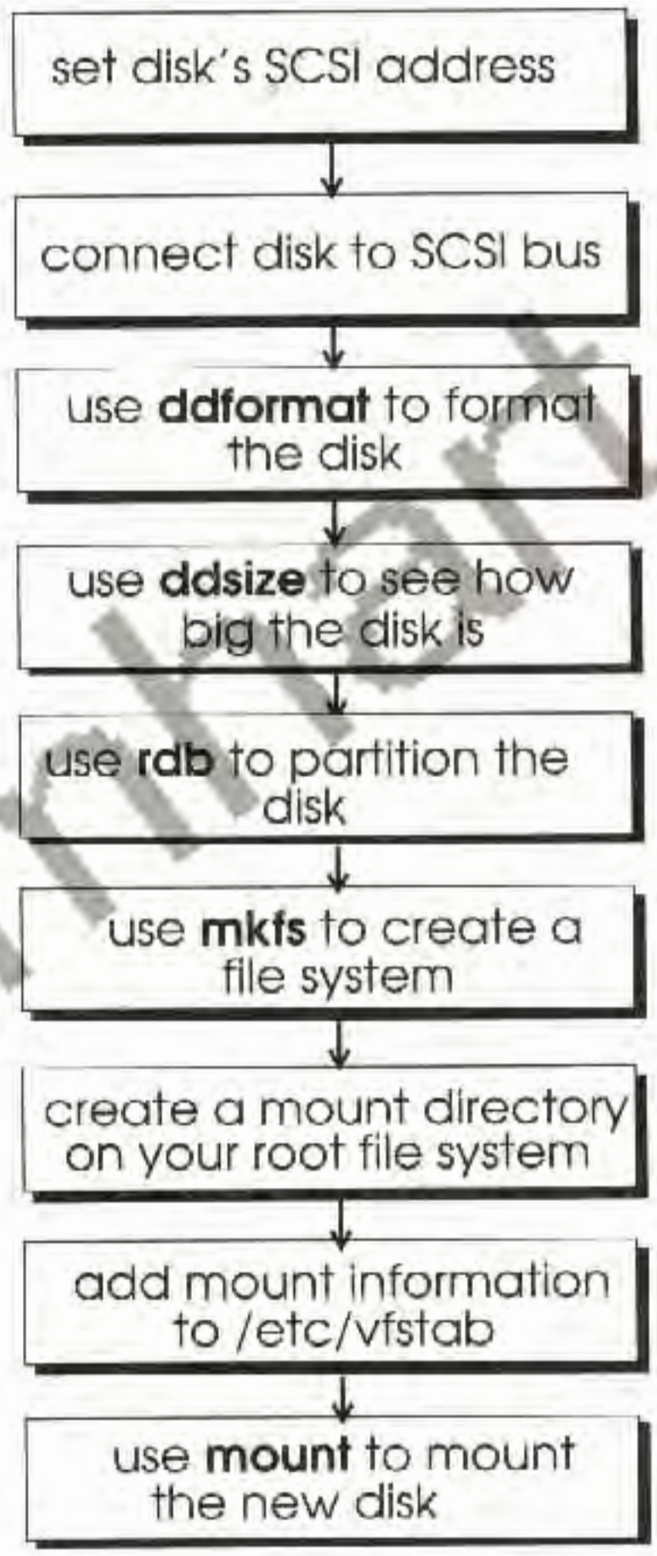
Adding a hard disk

What is a SCSI disk?

SCSI stands for Small Computer Systems Interface. Tape and hard disk drives are common examples of SCSI devices. You connect SCSI devices to your system simply by connecting a cable to the SCSI port. You can also link up to 7 SCSI devices together to form a chain.

What does it take to connect a SCSI disk?

There are nine steps to add a SCSI disk.



Change the address on the new disk

Each SCSI device must have a unique SCSI "address", and all of them must be turned on if any of them are to work. The following table lists the addresses already in use on a standard Amiga.

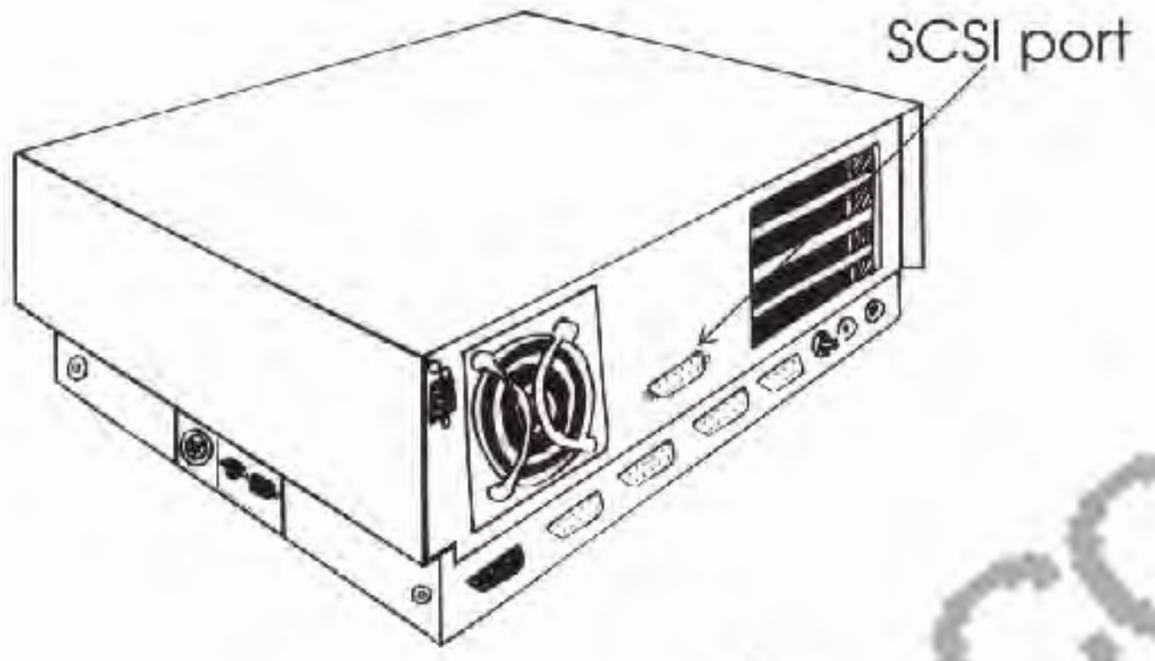
Address	Device that uses it
6	internal hard disk
4	external tape drive
7	SCSI disk controller

Make sure the new SCSI disk doesn't use one of these numbers or any others already in use.

You may need to set some jumpers on the disk to change its address. Check the disk manufacturer's instructions for setting the SCSI address. Use 1 if this is the first external SCSI drive.

Connect the SCSI cable

The 3000UX has a built in SCSI port. You can connect any external SCSI drive to this port.

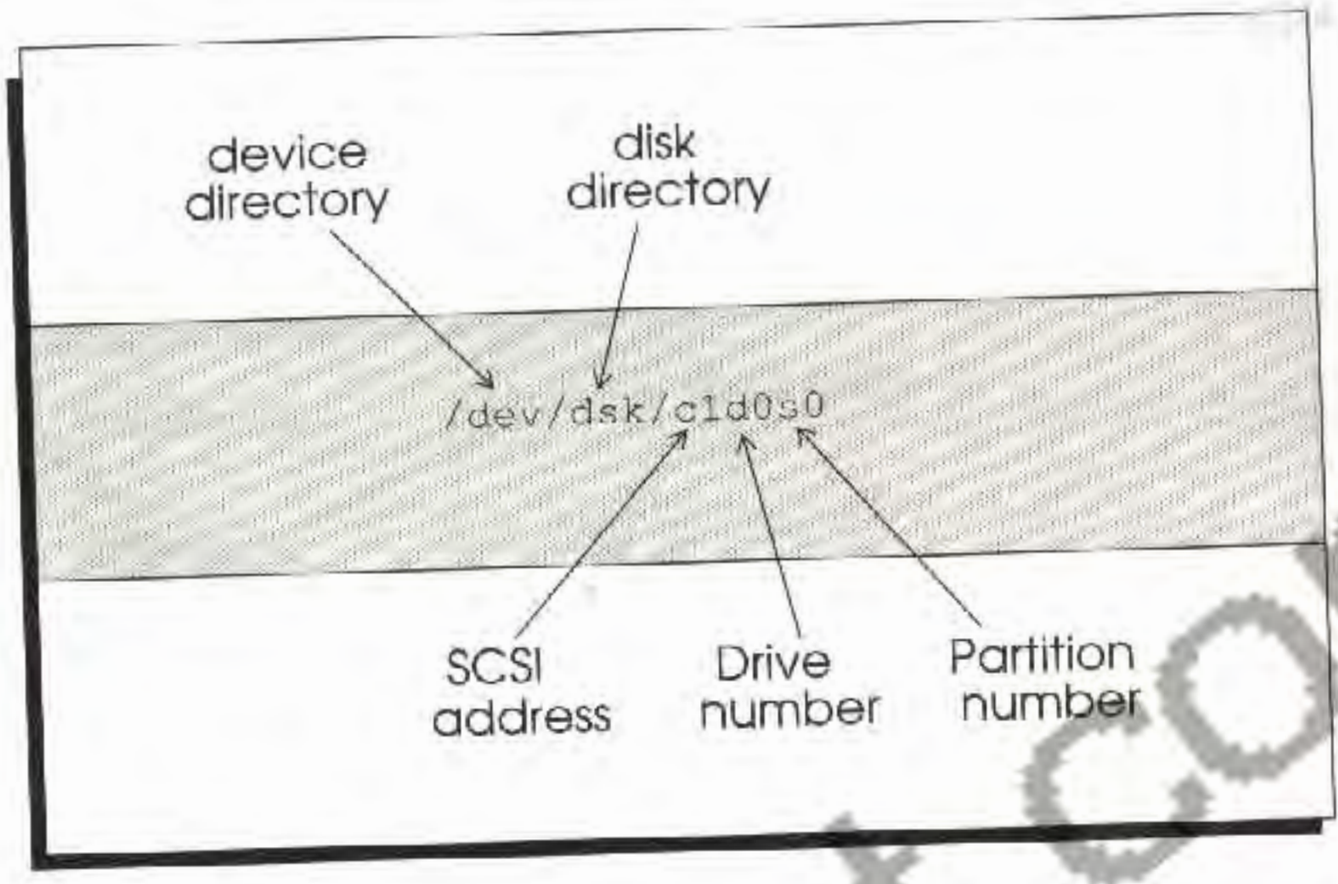


Give the disk a device name

Amiga UNIX provides many predefined disk device files in /dev/dsk.

SCSI disk c6d0s0 is your default hard disk. It has at least three partitions on it: Unix_Root, Unix_Swap and Unix_Boot. Each letter in the device name refers to a specific feature of the disk. The c refers to the SCSI device number (or address), the d refers to the disk number, and the s refers to the partition number. Thus, if your internal disk is c6d0s0 with 3 partitions, the partition names are c6d0s1, c6d0s2, c6d0s3 respectively.

Parts of a device name



Any device file with an s0 (like c4d0s0) refers to the entire disk. Any device with an s1-s7 (like c4d0s3) refers to a partition on the disk.

Fields	What does it do?
device dir	directory off root called /dev
disk dir	directory off /dev called /dsk
SCSI address	whatever you set the jumpers to identifies the drive
drive number	identifies the drive
partition #	specific number for each partition

Use ddformat to format the disk

If your disk isn't already formated, you should do so using **ddformat**. To format a disk, type **ddformat** followed by the name of the device you are formatting.

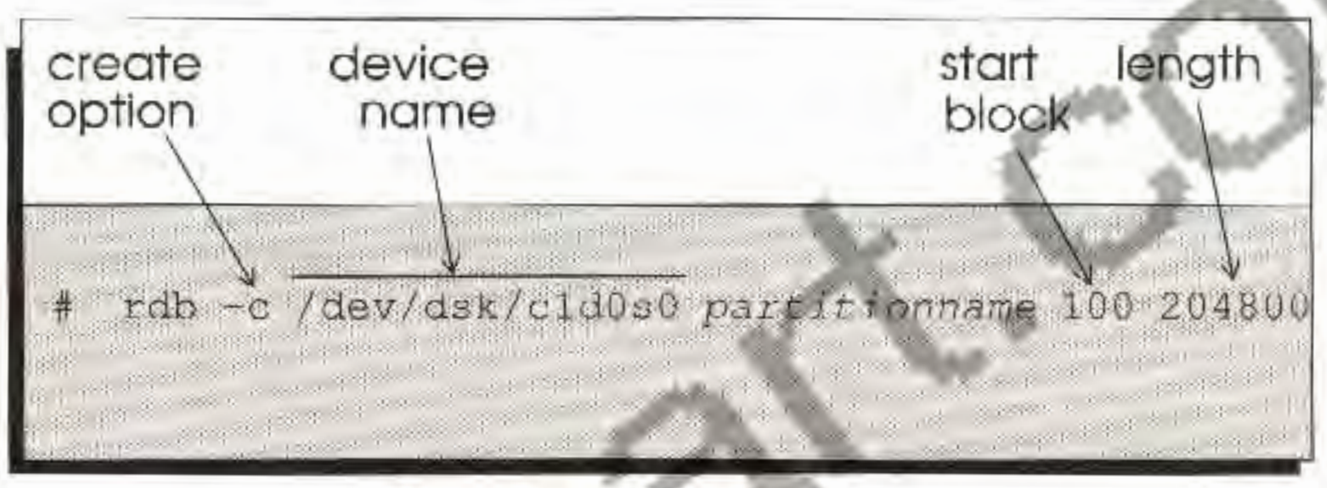
```
#  ddformat /dev/dsk/c1d0s0
```

Use rdb to create and show partitions

You use **rdb** (rigid disk block) to perform three disk partitioning tasks

- display information about the current partitions (**rdb** ***devicename***)
- create a basic partition
- add a variety of custom formatting features to a hard disk partition

The following example shows how you might put one partition on a 100 megabyte disk.



Option	What does it do?
-c	tells rdb to create partitions; always create using s0 (the whole disk)
dev name	file in /dev/dsk for this disk
partitionname	type a name for this partion
start block	point on disk where partition starts; leave about 50K (100 blocks) free at beginning of the disk
length	length (in 512 byte blocks) of this partition. length = **ddsize** - start block

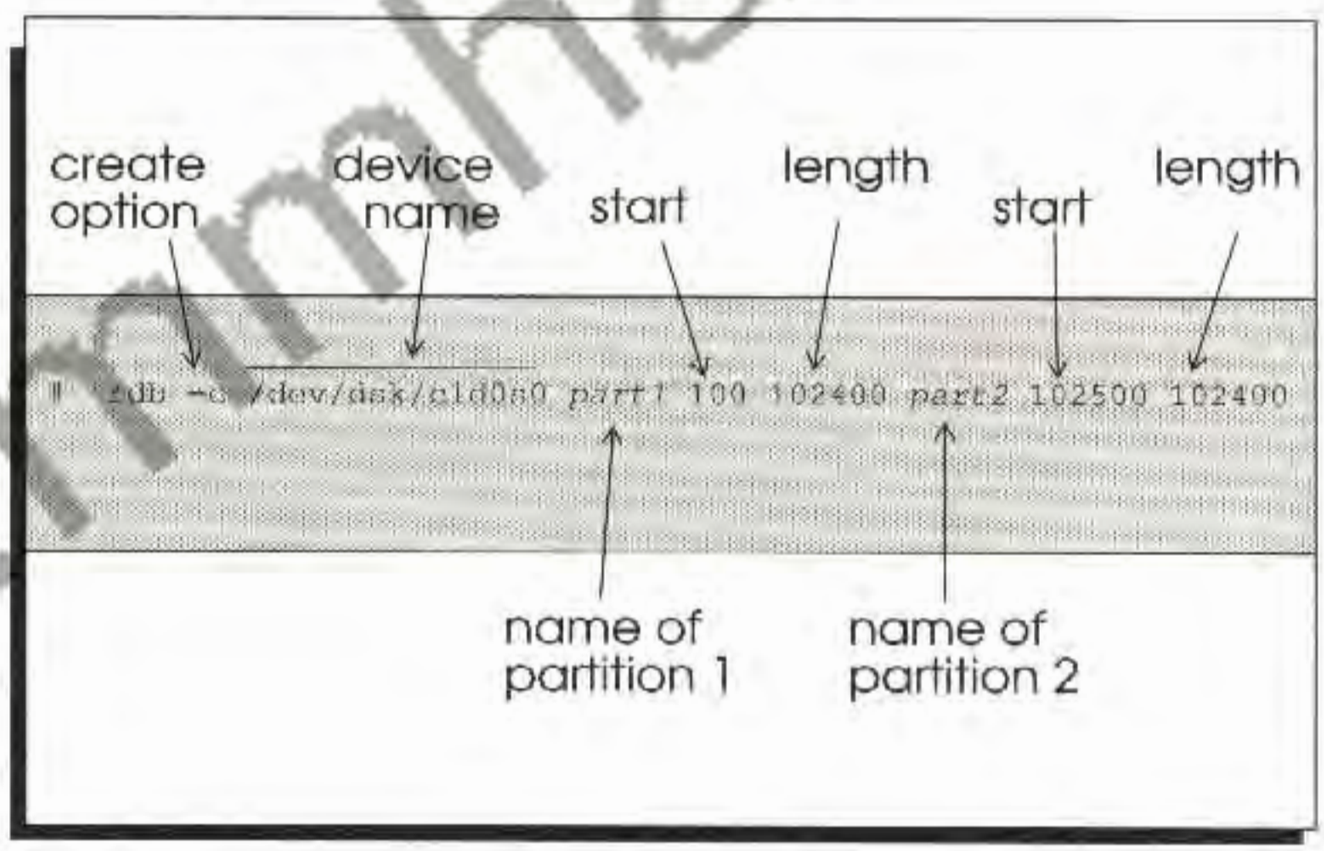
Make as many partitions as you need

Most Amiga UNIX computers ship with their disks divided into three partitions for Amiga UNIX. The first partition's device name is c6d0s1, the second is c6d0s2, and the third is c6d0s3. You can use the rdb command to set up partitions on your own disks.

NOTE: **rdb** destroys existing partitions, which means you can no longer access files on the disk. Use **rdb** with care.

You can only run the rdb command on the entire disk, which is c1d0s0 in our examples and c6d0s0 for the standard Amiga disk.

You can make two 50 megabyte partitions on the disk with one command.



The starting point of the second partition (102500) is the starting point of partition 1 (100) plus the length of partition 1 (102400). The length of partition 2 is the same as partition 1 in this case because you are dividing the disk in half.

Each partition you put on a hard disk is treated as a separate file system. You mount each partition (file system) on its own directory.

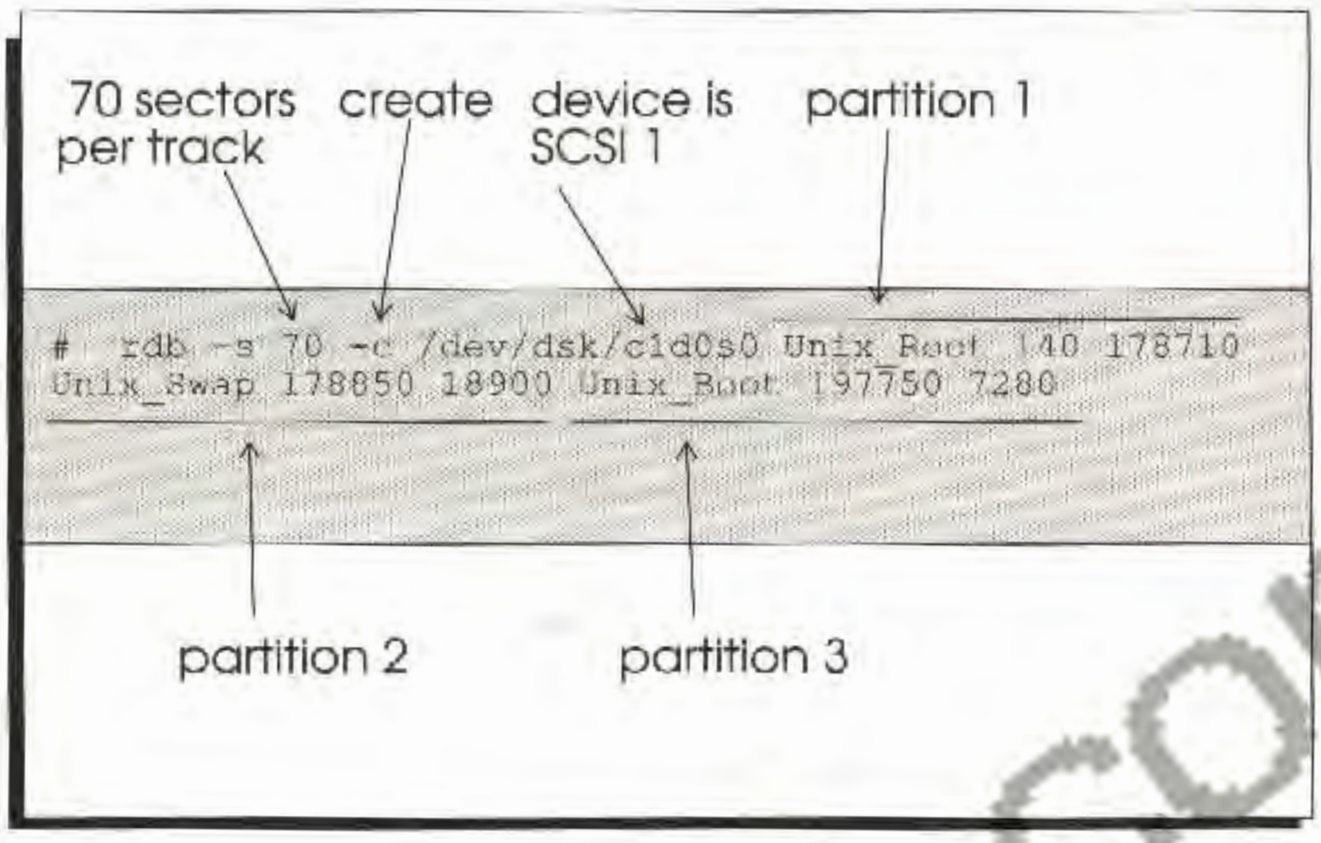
rdb options

The following table describes more rdb options.

Option	What does it do?
b	Make selected partition non-bootable
B	Make selected partition bootable
c	Create a new rdb
C n	Set number of custom boot blocks on selected partition to n
F n	Set file system type on selected partition to n
H	Display information about current partitions in a long format
p n	Select partition n
P n	Set boot priority on partition to n
s n	Set sectors per track to n

The following examples illustrate some more uses for **rdb**.

Three partitions and 70 sectors per track

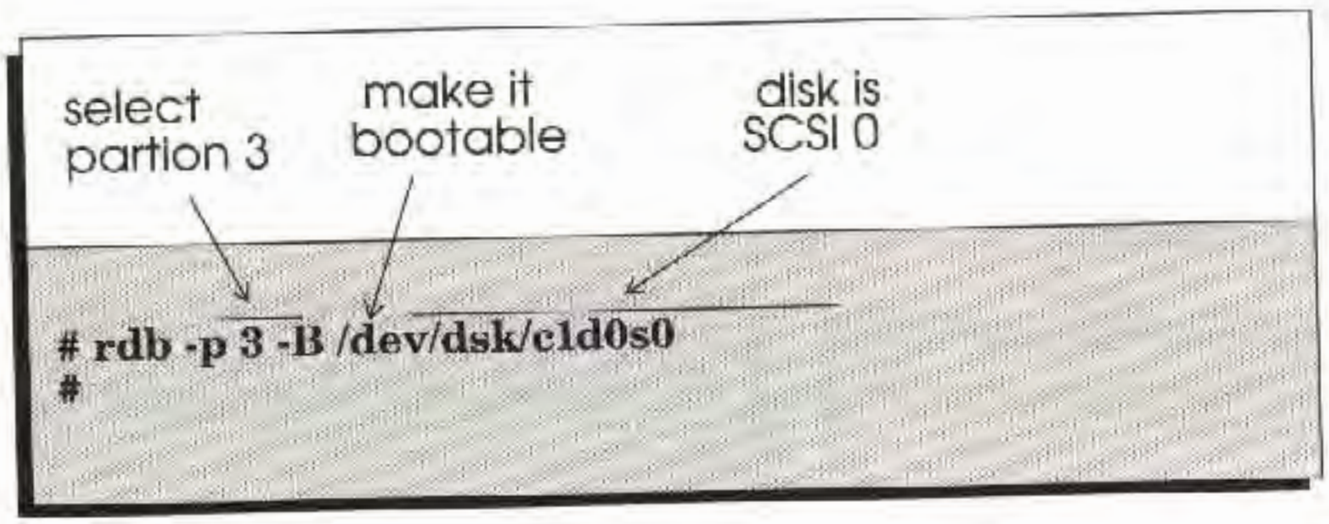


This example creates an rdb on SCSI disk 1 with 70 sectors per track and three partitions. Each partition is given a name, a starting position, and a size (in 512 byte blocks).

The previous command produces the following rdb list. Note that the settings in the list exactly match the instructions you gave **rdb**.

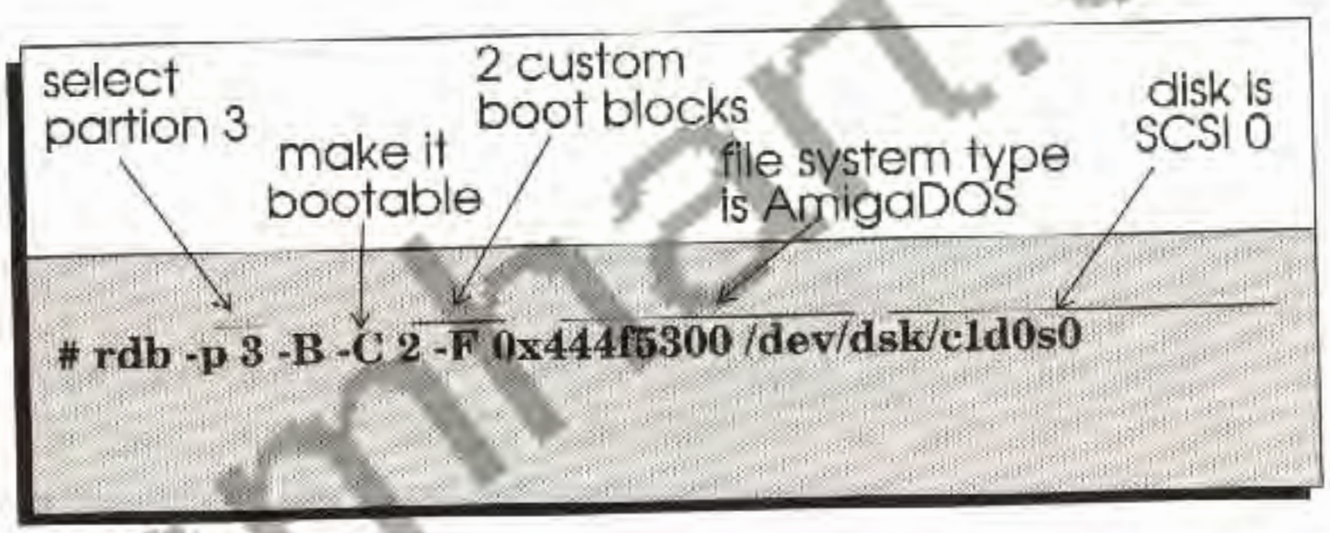
```
# rdb -H /dev/dsk/c1d0s0
#  Name        Start   Length    Size
1: Unix_Root     140    178710    87 meg
2: Unix_Swap     178850  18900     9 meg
3: Unix_Boot     197750  7280      3 meg
#
```

Customize a bootable partition



This **rdb** command makes partition 3 on SCSI disk 1 a bootable partition.

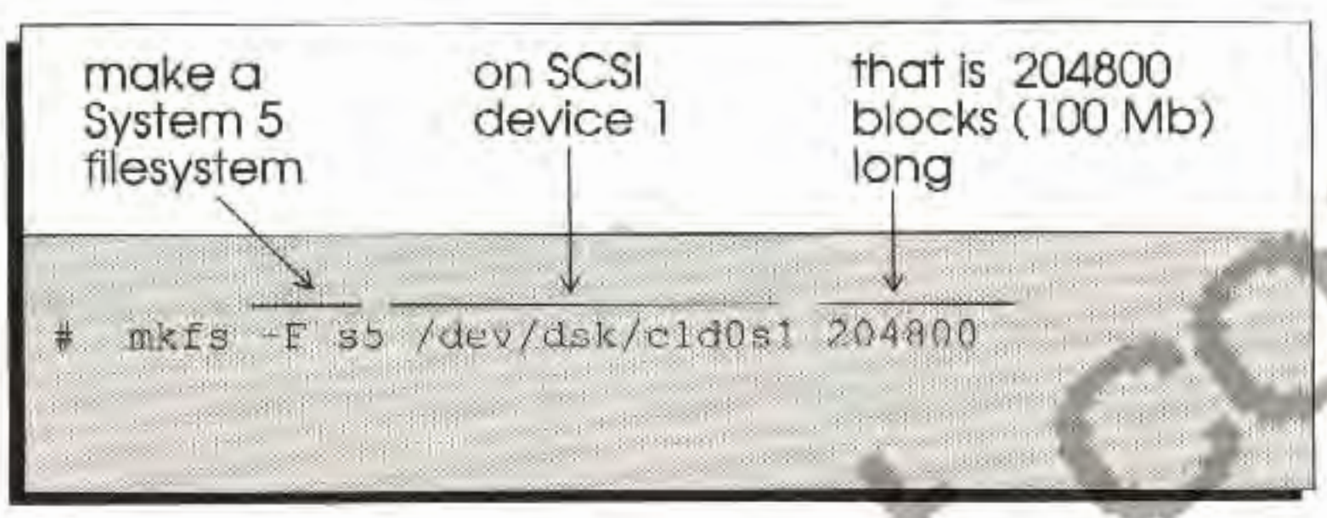
Define an AmigaDOS partition



This **rdb** example makes partition 3 bootable with 2 custom boot blocks and sets its filesystem type to DOS\0 (0x444f5300 is hex for DOS\0) for use with AmigaDOS.

Make a filesystem on the disk

Use the **mkfs** command to build a file system on an external disk (**ddformat** erases the disk; **mkfs** builds an empty file system on it). Run **mkfs** for each partition on your disk. You specify the partition size in 512 byte blocks. The following example shows what to type for a 100 megabyte disk with 1 partition.



NOTE: You should never run **mkfs** on s0 (the whole disk). You should only **mkfs** on partitions (s1-s7). Likewise, you should never run **rdb** on anything but s0. **mkfs** works on partitions; **rdb** works on disks.

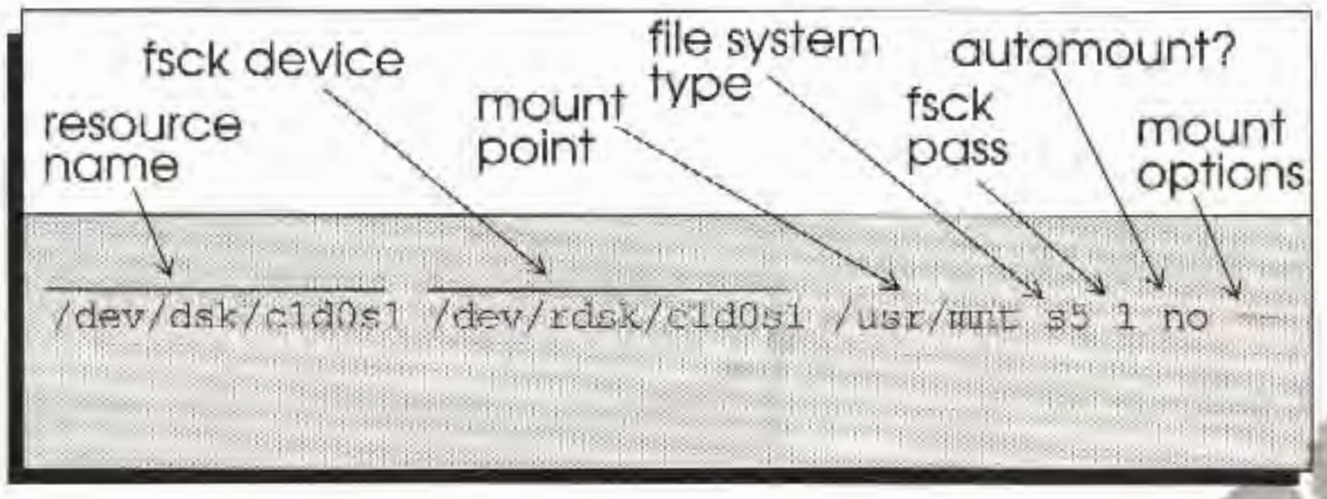
Create a mount directory on the internal drive

Create a directory on your main hard disk. Call it something like */usr/mnt*. This is where you are going to mount the new disk. If you mount the drive on an existing directory, anything in that directory will be covered, and is therefore unavailable until the disk is unmounted.

Add the file system information to /etc/vfstab. Once you put the line in vfstab, you can refer to the disk by its mount directory, rather than by its longer device

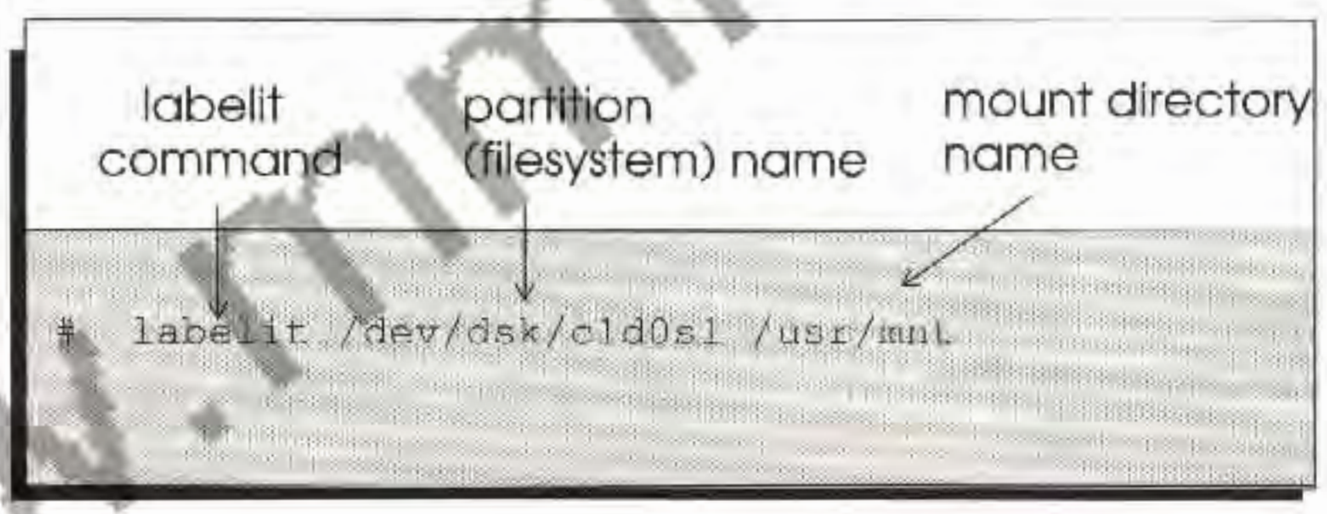
Define the file system in /etc/vfstab

name. Also, you can make the mount automatic, so the external disk partitions are mounted whenever the system boots.



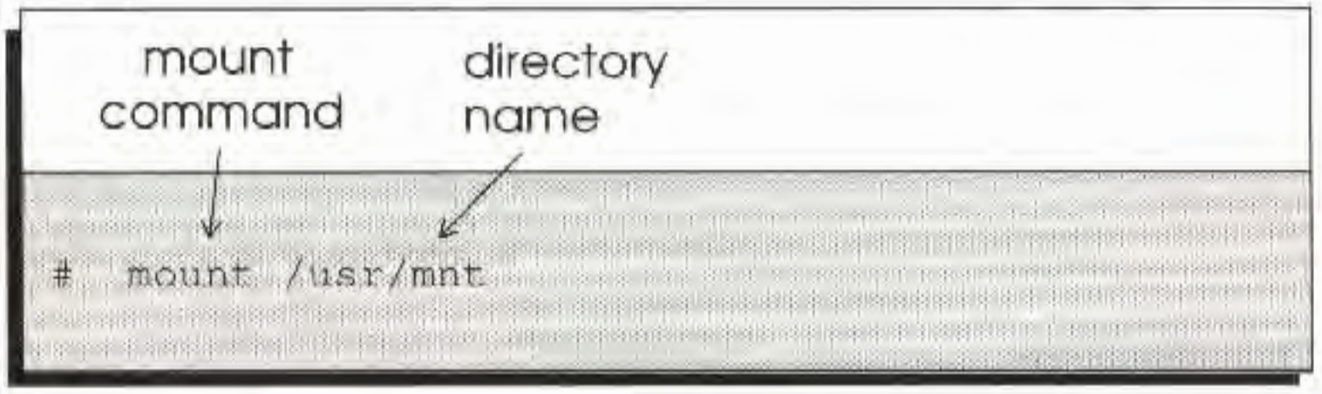
Run labelit on the filesystem

Labelit lets you label your new filesystem. The label prevents the **mount** command from giving you an error message when you mount the new filesystem. You use the mount directory name as the label.



Mount the disk

Use the **mount** command to mount the disk.



mount reads /etc/vfstab, finds the line for this directory, and mounts it.

After you mount the disk, you can **cd** into the mount directory, just like any other directory, and start using the disk. This is what "virtual file system" means; the system does not care what disk drive or disk partition you are using. You simply choose files and directories; it keeps track of where they are.

Expand the lost+found directory

The lost+found directory on your mounted partition is reserved for use by fsck to save corrupted files. (Most users do not need to understand lost+found; damaged files are automatically thrown away during shutdown and reboot). Experienced administrators can try to recover lost+found files, but it is not an easy process.

If you think you might want to use lost+found, expand the empty directory by running **mklost+found** ***directoryname***.

```
# mklost+found /usr/mnt
```

Using the UNIX shells

Shell scripts

You can use shells to run a sequence of instructions automatically. The sequences are actually executable files called "shell scripts". Shell scripts use shell commands and common programming language constructs (like *if*, *else*, *do*, *while*, *break*, and *case*). To run a shell script, type the name of the script.

Start-up files

Examples of shell scripts are .profile (run by ksh) and .login (run by csh); both are start-up files you can create in your home directory. The system start-up file is /etc/profile, but you can create a start-up file of your own by copying /etc/profile into your home directory as .profile (or .login for csh) and editing it to suit your needs. Each time you log in, the shell will set up your computing environment by reading the start-up file.

Shell and environment variables

A start-up file contains shell and environment variables which define how your terminal environment should behave. Shell variables are used by shell programs, shell scripts, and the shell itself. For example, one shell variable, PS1, determines what your prompt looks like. You can define the prompt as anything you want: a dollar sign, your system's name, your current path, or something else. Any variables that are used by shell programs (like **vi** and **mail**) are also called environment variables. A chart on the next page lists some of the most common shell variables.

Shell variables

Variable	Description
ENV	define the pathname of the file containing alias and set-up commands
HISTSIZE	define the number of previously executed commands kept in the command history file
HOME	define the pathname of your login directory
LOGNAME	define your login (username)
PATH	define the search path for finding and executing commands
PS1	define your shell command prompt string; the standard command prompt for the Korn and Bourne shells is $; for the C shell it is %.
PS2	define your shell input prompt string; the standard input prompt for all shells is >
SHELL	define your login shell
TERM	tell the shell what kind of terminal you are using

Display variables

Display shell and environment variables with the following commands:

Command	Description
echo $*variable*	display the value assigned to a shell or an environment variable
env	display the current environment variables and their settings only
set	display the current environment and shell variables and their settings (Korn or C shell)
setenv	display the environment variables and their settings only (C shell)

Change variables permanently

You can define variables permanently or temporarily for a session. To change a variable permanently, edit the .profile or .login files or create a shell script that you can run when you want the variable to change.

Change shell variables temporarily

To define shell and environment variables temporarily in ksh and sh, use the following commands:

Command	Description
variable=value	assign a value to the specified shell variable
variable=value **export** ***variable***	assign a value to the specified variable and export the variable to the shell environment

To define shell and environment variables temporarily in csh, use the following commands:

Command	Description
set ***variable=value***	assign a value to the specified shell variable
env ***variable=value***	assign a value to the specified environment variable
setenv ***variable=value***	assign a value to the specified variable and export the value to the shell environment

Refer to the **env**, **set**, and **setenv** *Command Reference* pages for more information.

History file commands

The Korn and C shells maintain a history file of commands you previously executed. The number of commands kept in this file depends on the value assigned to the HISTSIZE environment variable. You can display your command history file and execute commands listed in the file using the following commands:

Command	Description
history	display previously executed commands
r **r *nn*** **r *command***	execute the previous Korn shell command execute the command referenced by command line number *nn* in the Korn shell history list execute the most recent command in the Korn shell history list
!! **!*nn*** **!*command***	execute the previous C shell command execute the command referenced by command line number *nn* in the C shell history list execute the most recent command in the C shell history list

Processes

Each background process you run is called a job. When you create a background process in the Korn or C shell, the shell assigns to it a job number (enclosed in square brackets) and a process ID (PID) number. The ksh, csh, and jsh shells allow you to terminate a job, stop a job, and control whether it runs in the foreground or background. These shells also let you list your active background jobs.

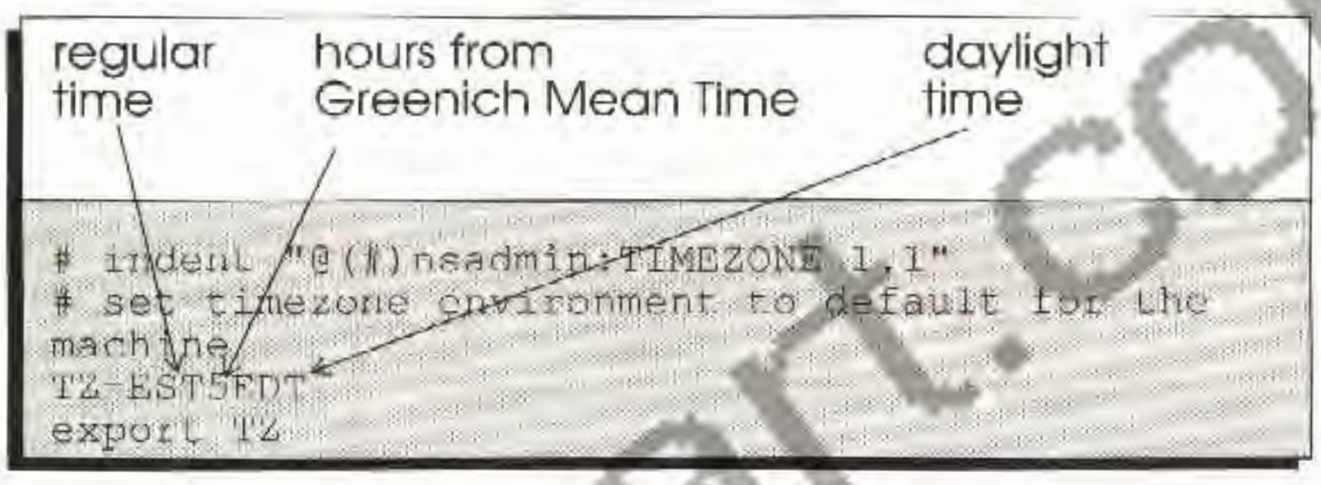
Command	Description
jobs	list active background processes under jobs control
kill *%n*	terminate job number *n*
CTRL-Z	stops a job that is running in the foreground
fg	place a job that is running in the background into the foreground, or restart a stopped job and runs it in the foreground
bg	place a foreground job that has been stopped into the background
%n	place background job number *n* into the foreground
%n **&**	place foreground job number *n*, which has been stopped, into the background

Configuring your system

Change the time and date

The installation tape sets up a standard UNIX system, but there are many ways to customize both the system and your individual user account. A few common customization options are summarized in this section.

To set the timezone, edit /etc/TIMEZONE. The /etc/TIMEZONE file contains a variable that you set to be your default timezone. This variable includes the abbreviation for regular and daylight time and the hours from Greenwich Mean Time.



Timezone	Values for TZ		
Greenwich	GMT	0	
Europe	MEZ	-1	
Atlantic	AST	4	ADT
Eastern	EST	5	EDT
Central	CST	6	CDT
Mountain	MST	7	MDT
Pacific	PST	8	PDT
Yukon	YST	8	YDT
Alaska	AST	10	ADT
Bering	BST	11	BDT
Hawaii	HST	10	

To change the date and time, use the **date** command.

```
#  date mmddHHMMyy
```

If you don't specify the year, **date** assumes you mean the current year. You can also type **date** without any arguments to see the current date and time.

Secure the system

When you first started Amiga UNIX, you had the option of assigning a password to the default login accounts. If you didn't assign one then, you can do so now by running a program called **passwdall**. This program asks for a password and secures every system account with this password. If you want to remove the password from all system accounts, type **passwdall -d**. You must be root to run **passwdall**.

Customize your login environment

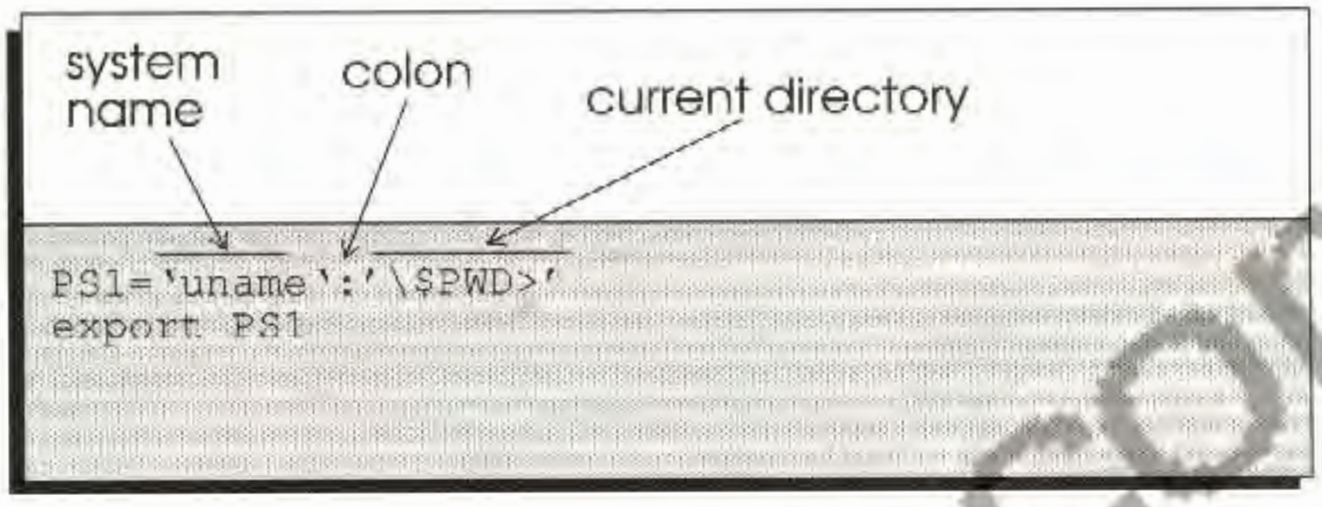
When you first start Amiga UNIX, you are working in the default environment we set for you. You can change your default environment by creating and modifying a file called .profile in your home directory.

Every time you log in to your computer, your login shell looks for a startup file in your home directory. It contains information specific to you, such as what your prompt should look like and whether you want OPEN

LOOK to start automatically. The Bourne and Korn shells use .profile; the C shell uses a file called .login. Both serve the same purpose.

Make a custom ksh prompt

If you use ksh, put the following lines in your .profile.

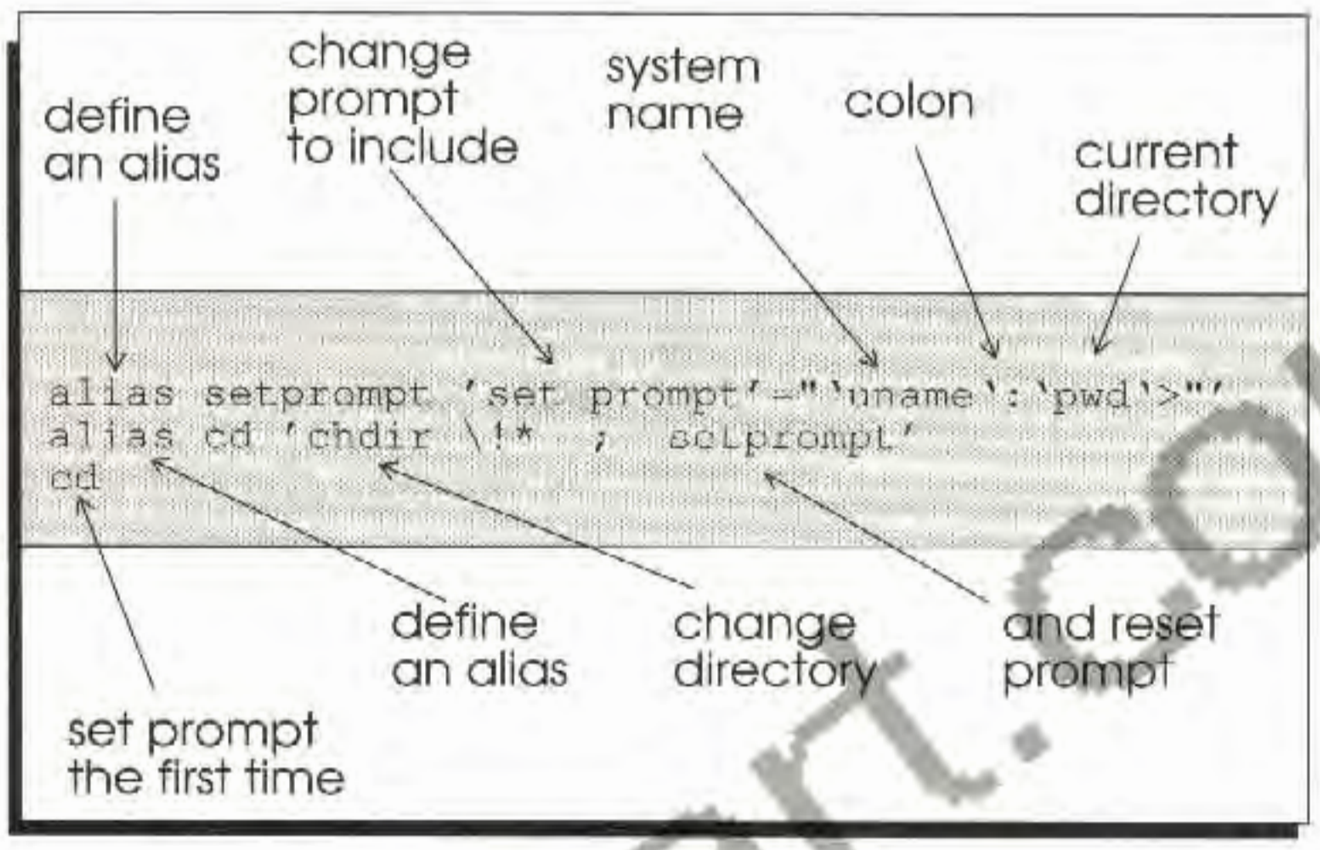


These lines make your prompt display your system name and current path, as in the example below.

```
amiga:/home/joe>
```

Make a custom csh prompt

You can also make the same custom prompt in the C shell, by putting slightly different commands in your .login file.



Set up aliases

You can assign an alias that invokes any UNIX command. You define aliases in your .profile (or .login for C shell). For example, you could create an **lt** command that always displays the long format by time (**ls -lt**).

```
alias lt='ls -lt'
```

Virtual screens

You can change the look of each virtual screen by changing its font size and background color. You change the virtual screens by editing lines in /etc/inittab. See *Editing system files* for help editing /etc/inittab.

Install optional parts of Amiga UNIX

The tape that came with your Amiga computer is divided into four parts:

- Amiga UNIX
- X Window System development tools
- complete AT&T documentation for Release 4
- public domain source code

Use cpio to copy the optional files

The only part that is installed automatically is Amiga UNIX. You can install the other three if you need them. You need about 55 megabytes of free disk space to install all the optional files. See *Installing Amiga UNIX* for more information.

Adding terminals to your system

How many terminals can you add?

With a normal Amiga UNIX license, you can have two users logged in to the system at a time. The other user can be over a network, using your virtual screens, or on a terminal attached to your Amiga. The terminal can be another Amiga, a PC, a dumb terminal, or a different kind of computer. If you want more than one terminal, you can add a serial port expansion card to your system.

Add a terminal to the serial port

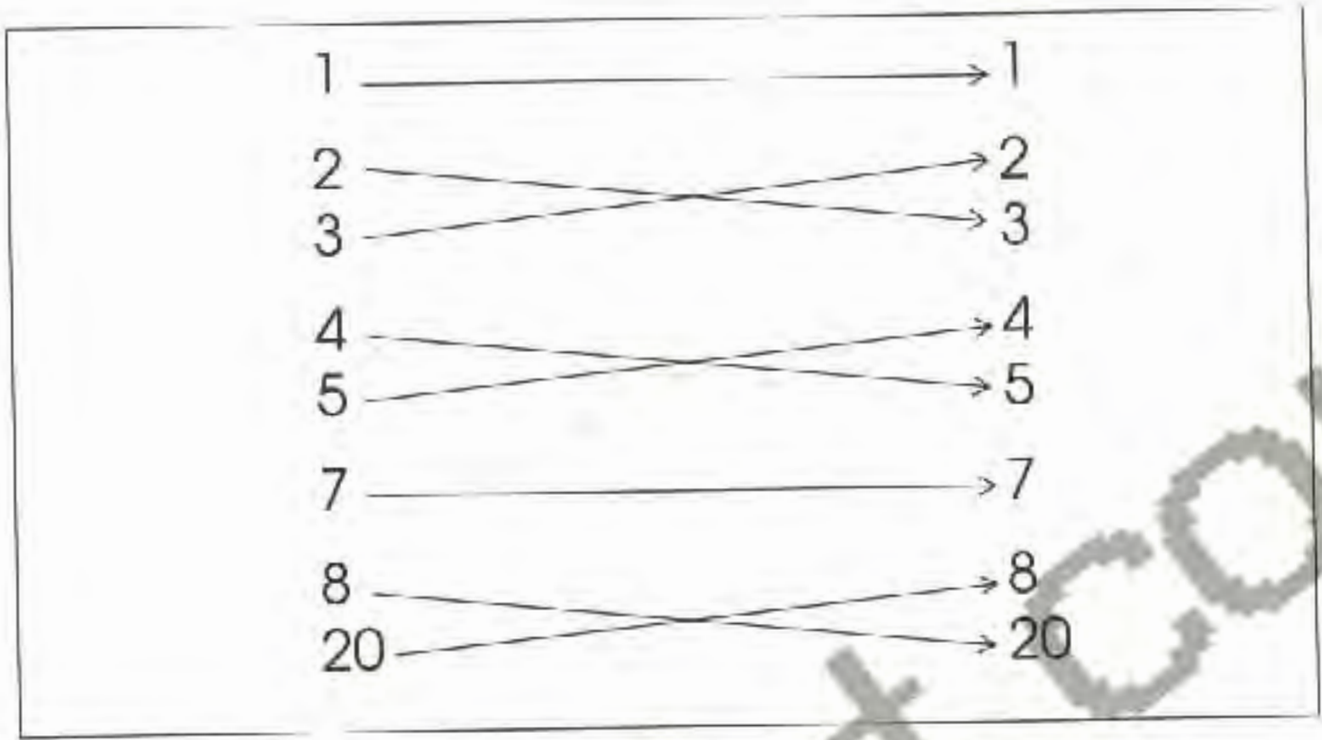
You need to do three things to add a terminal:

- connect it with a cable to the serial port
- change the port setting in /etc/inittab
- change the settings on the terminal to match the Amiga serial port

You can then use this terminal just like your monitor, except it will not have virtual screens.

You need a serial cable

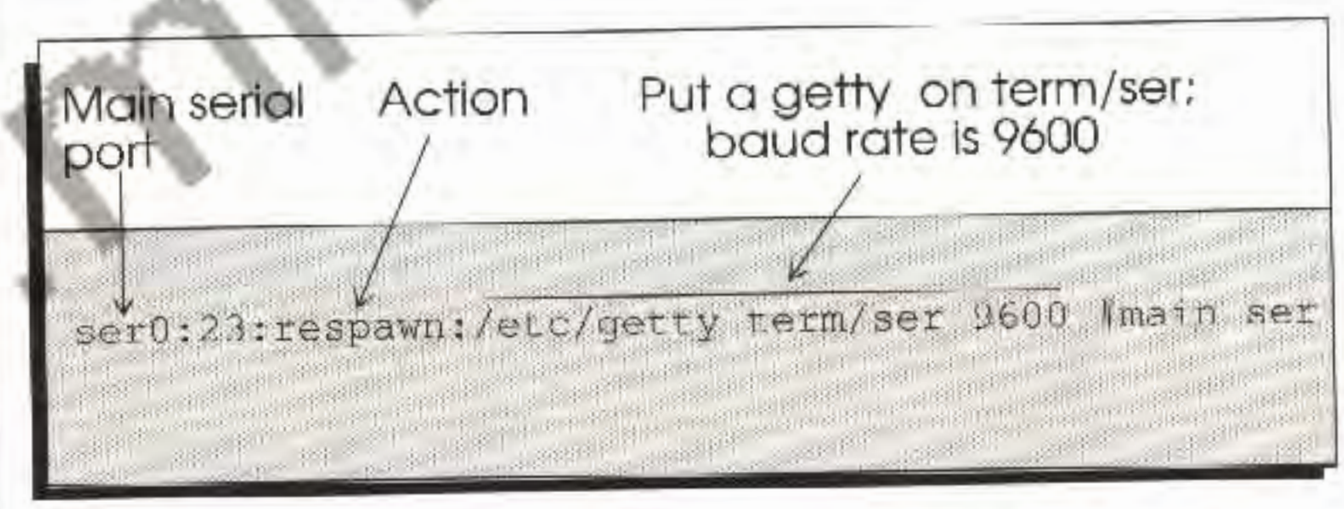
You can make or buy a 25 pin serial cable. Only a few of the pins are necessary. This diagram shows the configuration for the cable.



Configuration for serial cable

Change the serial line in /etc/inittab

You need to set the baud rate and terminal name in /etc/inittab and change the action from **off** to **respawn**.



See *Editing system files* earlier in this book for more information.

NOTE: After you make changes to /etc/inittab,type **init q** at the command prompt. This notifies **init** to check the new settings.

Match terminal settings

The hardest part of setting up an extra terminal is getting the cable and terminal settings right. If you use the configuration on the previous page, you shouldn't have trouble with the cable. Setting the terminal correctly to match the computer cable could take a while.

Terminals vary from manufacturer to manufacturer. If you have trouble with the terminal settings, contact the company that manufactured the terminal. You won't get a login prompt on the terminal until all settings are correct.

Shutting down and restarting your computer

What is shutdown?

You should never just turn off or reboot your Amiga UNIX computer. Always run the **shutdown** command first. **shutdown** brings your system down safely by:

- letting people know the system is coming down
- killing running processes
- saving all current memory blocks to disk
- switching the system to single user mode.

Also, always run **shutdown** before doing any maintenance on your system, such as adding disks or connecting tape drives. If you don't run **shutdown**, the current activity never gets closed or saved properly, and you may lose disk files during the **fsck** cleanup when you reboot.

Who can run shutdown?

Only root can run **shutdown**.

How do you run shutdown?

You shutdown your system by typing the **shutdown** command, using any or all of 3 options: skip prompts (-y), delay in seconds (-g), and init state (-i).

```
#  shutdown -y -g delay -i initstate
```

Option	What does it do?
-y	skip confirmation prompts
-g *delay*	seconds before shutdown occurs
-i *initstate*	lets you halt the system completely (0), go into single-user mode (S) for maintenance, or reboot (6)

Normally, simply typing **shutdown** and using the default options is sufficient.

What happens after you start shutdown?

After the delay interval, the **shutdown** command without any options brings the computer down to single-user mode. You can do one of three things at this point:

- type CTRL-D to restart the system
- log in as root
- turn the power off

Restart your system

If you log in as root, most likely to do maintenance, your console screen is the only available screen. When you are ready to restart multi-user mode, type **init 2**. Virtual screens won't work in single-user mode.

Backing up your files

Where should you put the backup?

You should get in the habit of backing up your important files by making duplicates of them. You can put the duplicates somewhere else on your hard disk, on a floppy disk, or on a tape. To completely protect your files, you should keep copies of the files in more than one place.

How do you make backups?

You use the **cpio** command to make backups. By varying the options you use with **cpio**, you can write files (and whole directory structures) to a device, get a list of files on a device, and restore files from a device. That device can be a hard disk, a floppy disk, or a tape.

The rest of this section shows you how to use the **cpio** command to backup files.

If you plan to use floppy disks for your backup, you must first format them using the fdfmt command.

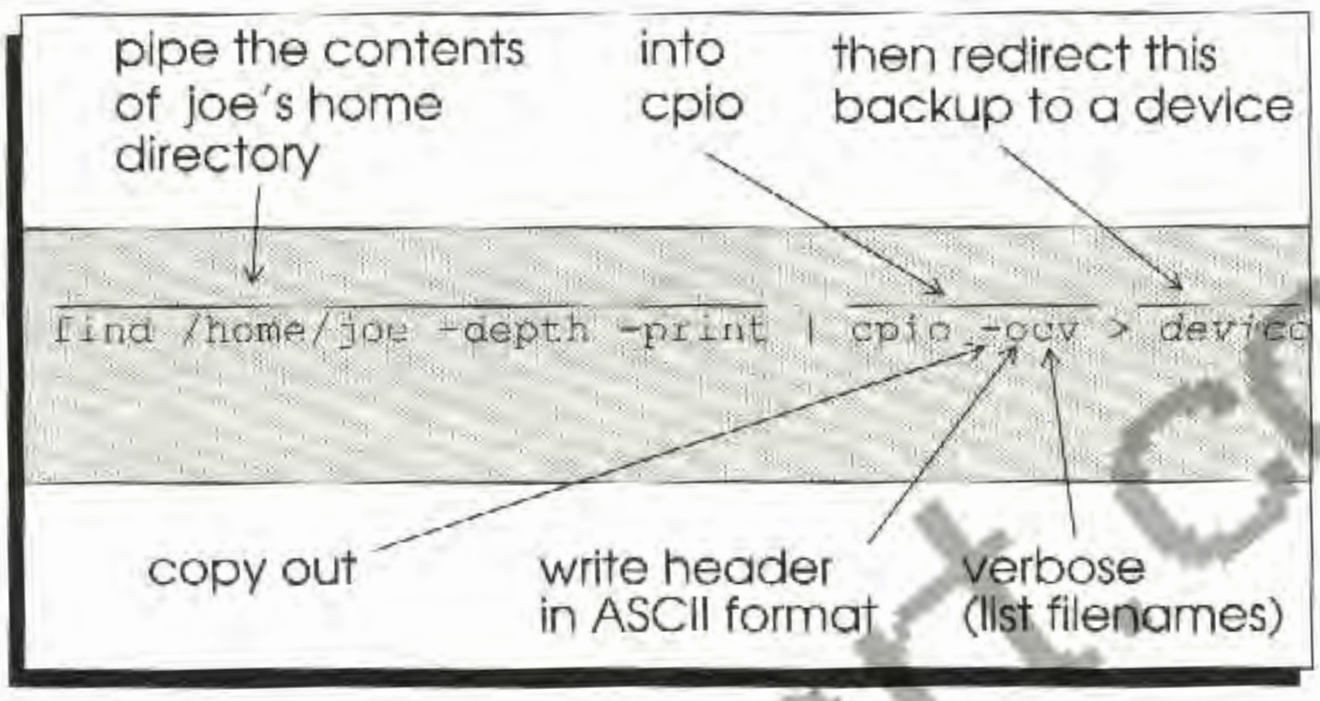
```
$ fdfmt > /dev/rdsk/fd0f
```

If you plan to use floppy disks for your backup...

If the backup requires more than on floppy disk, **cpio** prompts you to change the disks. Be sure you have enough formatted disks before you start.

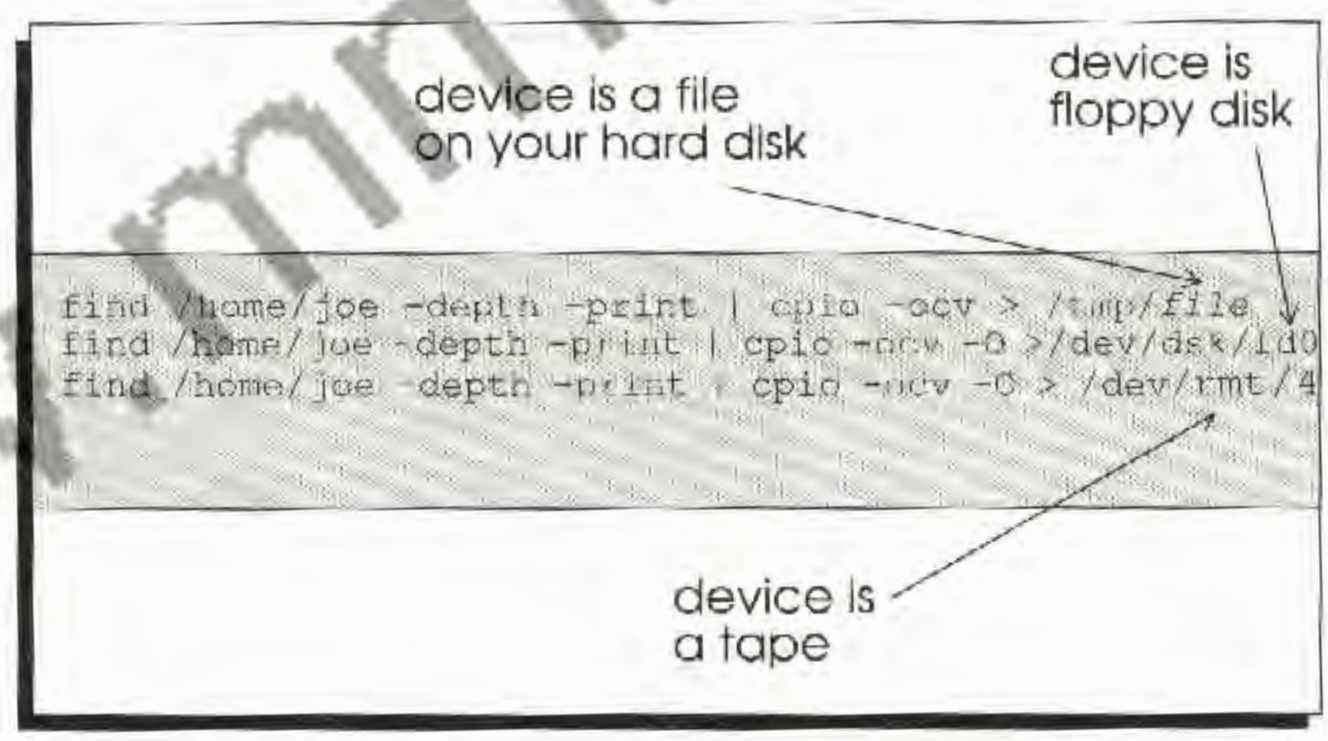
Back up files

The simplest way to save important files is to copy an entire directory structure. You use a **find** or **ls** command to list files, pipe the filenames into **cpio -ocv** (**cpio -ocv -O** for floppy disks and tape), then direct the output to your backup location.



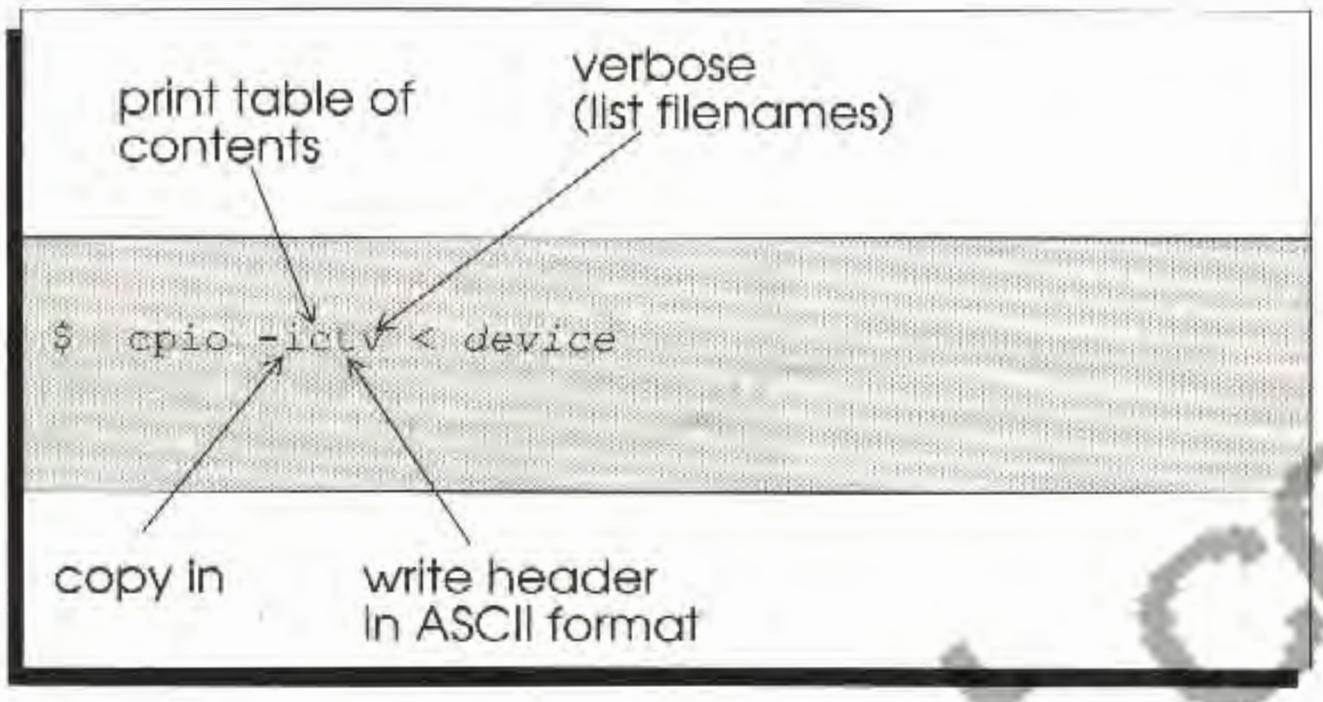
The following examples show what you type to backup files to a directory on a hard disk, a floppy disk, or a tape.

Examples of backing up files to hard disk, tape, and floppy disk



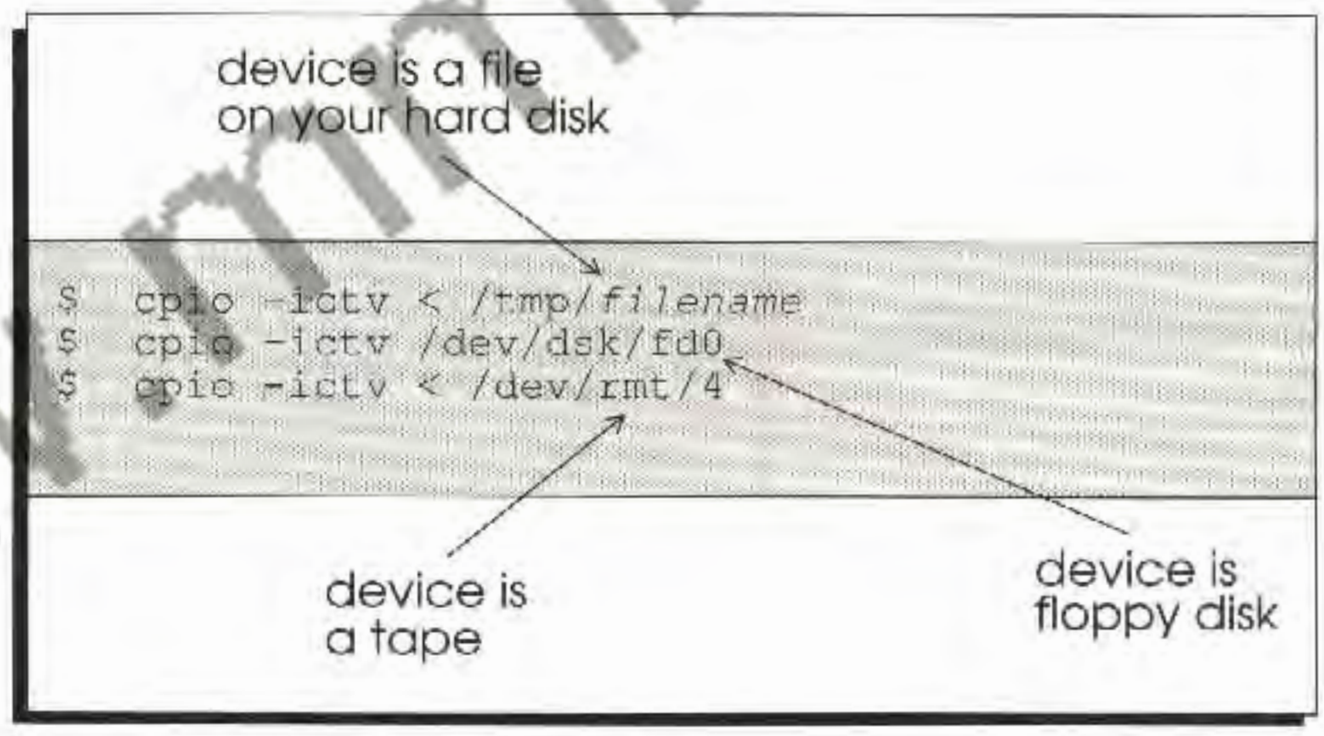
List the contents of your backup

You look at the contents of your backup by redirecting the output of a device (hard disk, floppy disk, or tape) into **cpio -ictv**.



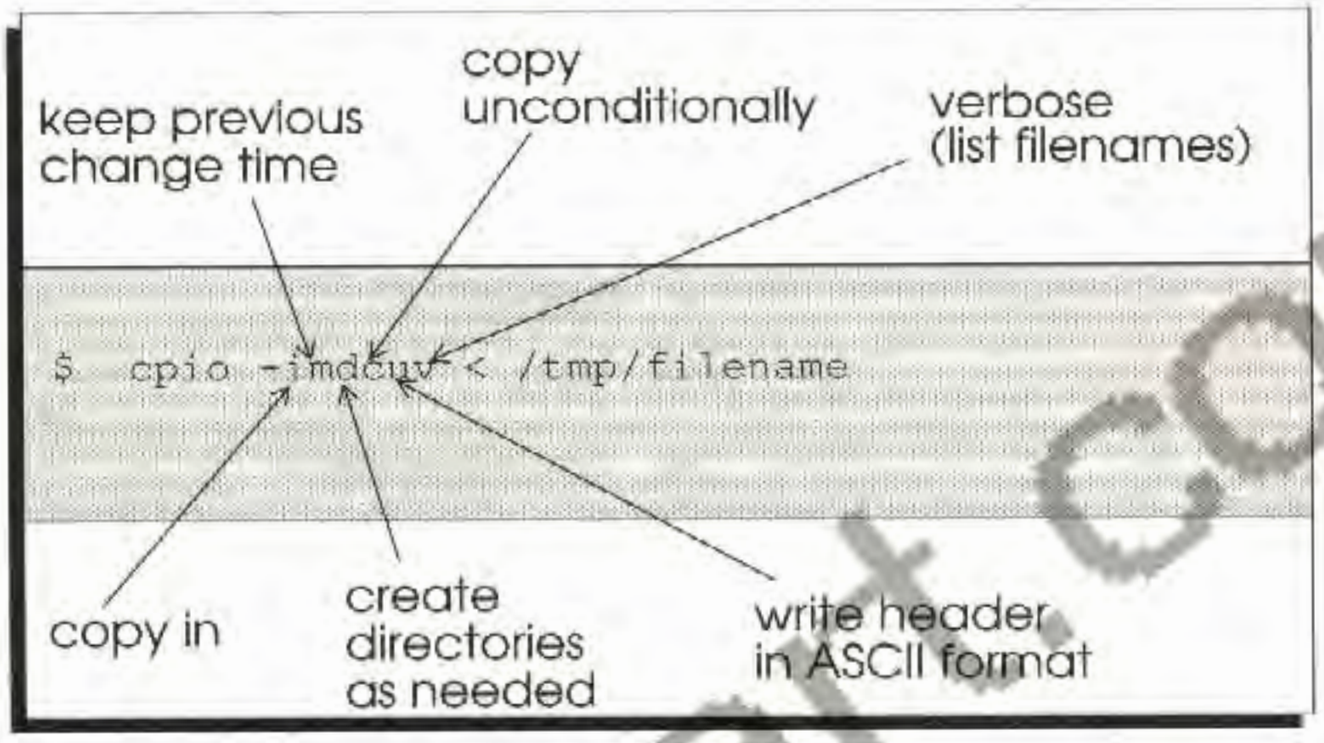
The following examples show what you type to look at the contents of your backup on a hard disk, a floppy disk, or a tape:

Examples of listing the contents of a hard disk, tape, or floppy disk



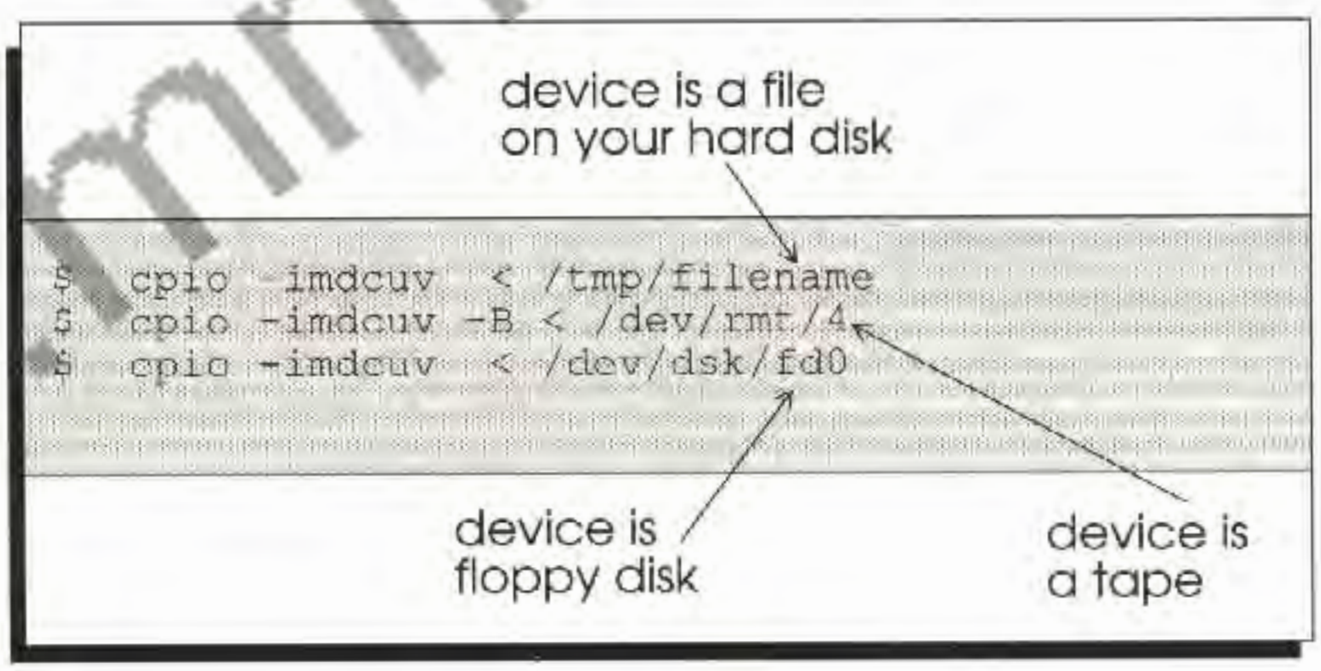
Restore files

The **cpio** command restores files and directories to exactly where it found them, overwriting anything currently there. You restore files by redirecting the output of a device (hard disk, floppy disk, or tape) into **cpio -imdcuv** (**cpio -imdcuv -B** for tapes).



The following examples show what you type to look restore your backup from a hard disk, a floppy disk, or a tape:

Examples of restoring files from a hard disk, tape, or floppy disk



Scheduling tasks using cron

What is cron?

The **cron** program runs commands for you at scheduled times. You can schedule processes to run when you aren't working on your computer, such as in the middle of the night, or at regular intervals during the day, such as every half hour.

What does cron do?

You could have **cron** automatically backup your home directory to another directory, display the time on your screen, or print your personal calendar. You can tell **cron** to do this on any given:

- month
- date
- day of week
- hour
- minute

How do you use cron?

The **cron** process is constantly working in the background, so you don't do anything directly with **cron**. Instead, you put the commands you want to execute and the times to execute them in a crontab file. The **cron** process reads the crontab and executes the commands at the specified times.

Who can create a crontab?

Any user can create a crontab. If you want to prevent users from creating a crontab, put their *usernames* in /usr/lib/cron/cron.deny. You must be root to edit /usr/lib/cron/cron.deny.

What are the default crontabs?

There are six default crontabs in /var/spool/cron/crontabs, containing scheduled maintenance procedures run by various system accounts:

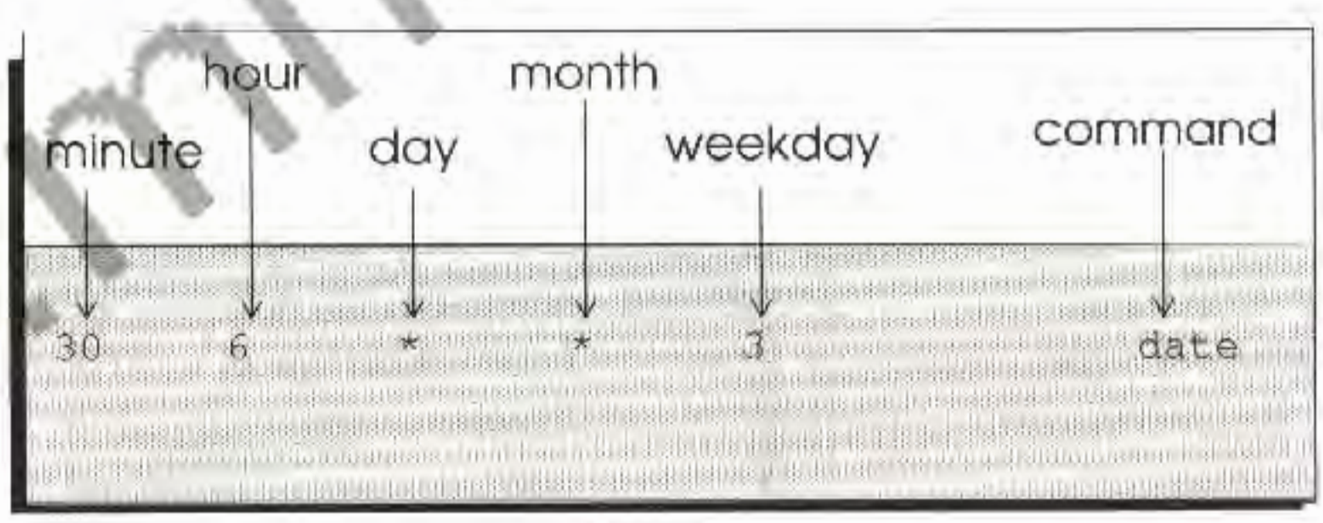
- adm
- lp
- root
- smtp
- sys
- sysadm

These crontabs are reserved for special system uses, and only root or the crontab owner can change them. After you create a crontab for yourself, it appears in this directory, with your username as its filename.

Create a crontab

You use the **crontab -e** command to create a crontab. After you type **crontab -e**, an editor appears into which you type the 6 crontab columns separated by tabs. Five of the columns define the scheduled time and the sixth is the command to be executed.

crontab format



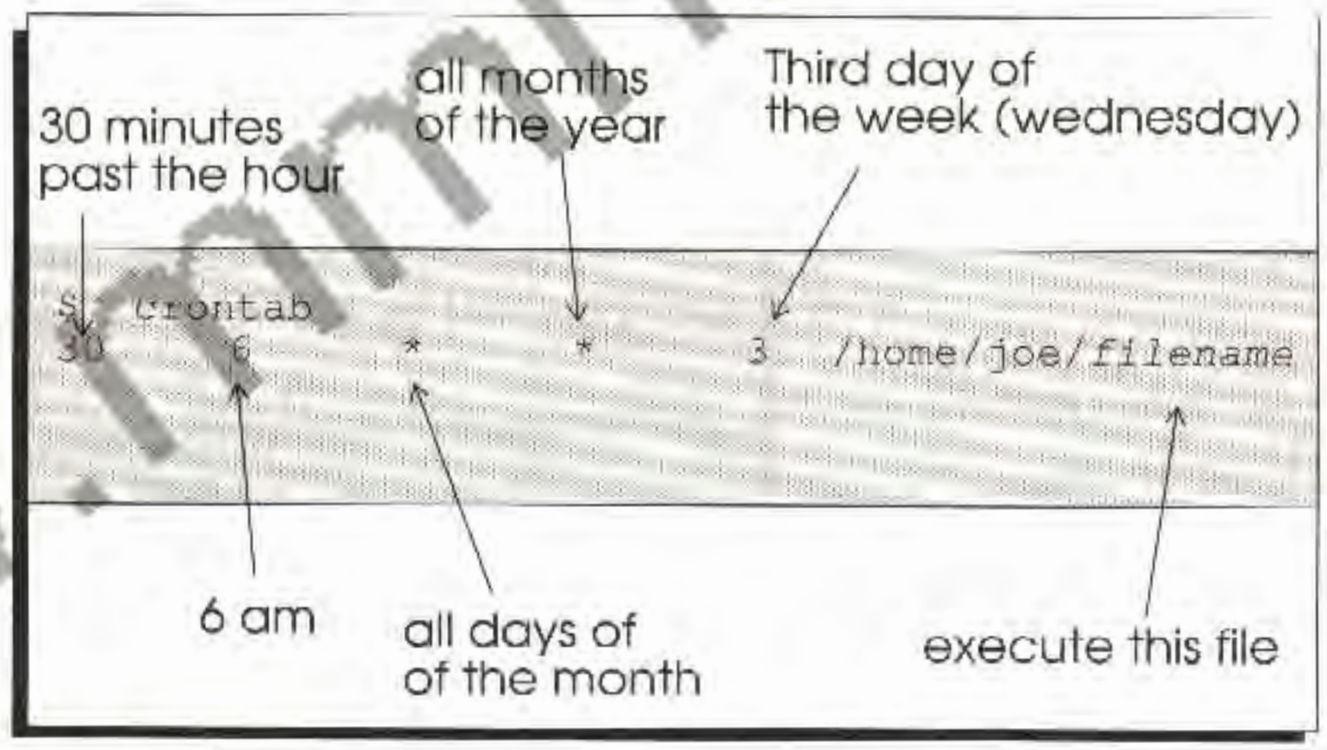
Field	What can you put in it?
minute	0 to 59 or * for every minute
hour	hour of day in 24 hour format (0-23) or *
day	day of the month 1-31 or * for every day
month	month of the year 1-12 or *
weekday	0 - 6 (0 for sunday) or * for all days
executable	command or executable file name

The day and weekday work together, one does not have priority over the other. You can schedule a job to run on the 12th of each month, Mondays and Fridays. If the 12th falls on a Tuesday, cron still runs the job.

Use the crontab command

The following example shows a crontab that executes *filename* at 6:30 AM every Wednesday. You can also put commands directly into the crontab.

Sample crontab entry



Edit an existing crontab

Use **crontab -e** to edit an existing crontab.

After you make your changes to the crontab, exit the editor as you would normally. If you made a mistake in the crontab, a message appears telling you so. You can edit the crontab to add lines, delete lines, and change lines.

Other crontab options

The following table lists two other useful **crontab** options.

Option	What does it do?
-l	List the contents of your crontab
-r	Remove or delete your crontab from /usr/spool/cron/crontabs

Experiment with the cron system

There are many ways to use a crontab. Experiment with the cron system by scheduling commands and your own executable files.

UNIX accounting files

What are the UNIX accounting files?

The UNIX system uses a complex accounting system for tracking things such as system usage by process or user, disk usage, and the amount of time a user is logged in. There are many other accounting files, many of which you will never have to use.

Accounting files accumulate data

Some of the accounting files are active by default, which means they are accumulating data and possibly using up valuable disk space. It's not important that you know how to use and interpret every accounting file, but it is important that you know how to find the ones that are active.

Periodically check the accounting files

If your disk starts getting full, you should check the size of these accounting files. If they are extraordinarily large, remove, empty, or edit them. The table on the following page lists the more common accounting files and their purpose.

The most common accounting files

Accounting file	file's contents
/var/adm/spellhist	get line when spell runs
/var/cron/log	cron's log
/var/saf/log	saf log
/var/saf/ttymon/log	port monitors log
/var/adm/sulog	tracks the su command
/var/adm/lastlog	last login information
/var/adm/wtmp	login history
/var/adm/wtmpx	login history
/var/spool/smtq/LOG	smtp (network mail) log
/var/lp/logs/lpNet	network printing log
/var/lp/logs/lpsched	print scheduler log
/var/lp/logs/request	print requests log
/usr/public/lib/news/log	Usenet news log
/usr/public/lib/news/errlog	Usenet news error

Some accounting files are automatically trimmed

Some accounting files are automatically trimmed by cron jobs. Most of these files are found in directories in /usr/spool/uucp.

Troubleshooting

Problem	Solution
Can't use one of the virtual screens	Make sure its action is set to respawn in /etc/inittab.
Can't use any of the virtual screens	Your system is in single user mode.
Can't kill a process	Make sure you typed the right process ID number (PID). Use the **-9** option with the **kill** command. The process doesn't belong to you. Log in as root, then try killing it.
Can't display other users' processes	Use the **-e** option with **ps**.
Can't open a system file for editing	Log in as root. Only root can change system files.

Problem	Solution
You created a new user account, but Amiga UNIX doesn't recognize it	Run **pwconv**. Every time you add a new user, you have to run **pwconv** to update /etc/shadow. Make sure you entered the line properly in /etc/passwd. Make sure there was no password entry in /etc/passwd before you ran **pwconv**, and that there is only an x in the password spot after **pwconv**.
Can't work with files after creating a new user account	Root created the user account, so root owns the directory. **chown** and **chgrp** the directory so that you (or the user) owns the directory.
You changed shells, now some commands don't work	Each shell is slightly different. **sh** and **ksh** are similar, but **csh** and **ksh** both have some features that **sh** doesn't share. The only way around this problem is to change back to the shell with which you are familiar.

Problem	Solution
SCSI disk doesn't work	The SCSI address is already in use. Set the device address to one that is not in use. Make sure you ran **rdb** only on s0 (the whole disk). Make sure you **mkfs** and **mount** only with s1 through s7 (partitions). The device name is wrong. Check the name again.
Can't make a serial terminal work	Baud rate is wrong in /etc/inittab, or port is still set to off (instead of respawn). Cable configuration is wrong. Terminal settings don't match your system settings.
Can't shut down your system	You aren't logged in as root on the console (F1). You specified the shutdown options incorrectly. You're not in the root directory (/).

Problem	Solution
Can't format or copy to a floppy disk	Make sure the write protect tab is closed.
Customizing aliases or prompts doesn't work	Log out then log back in. Be sure you're using the right startup file; .profile for **ksh** and **sh**, .login for **csh**.

UNIX command reference

File Commands

cat	display contents
cp	copy
more	display one page at a time
mv	rename or move
rm	delete
* ?	wildcards to match patterns in a file-

Directory Commands

cd	move to home directory
cd ..	move up one directory
find	search for a file
ls	list files in directory
mkdir	make a new directory
pwd	show current directory
rmdir	remove empty directory

Print Commands

cancel	stop a print job
lp	print
lpstat	printer status

Disk Usage Commands

df	disk space used and free
fdfmt	format 3.5" disk
tar	copy to, retrieve from
cpio	list files on disk or tape

Tasks You Do Using Amiga Unix Commands

You...	by using...
backup files	**tar** or **cpio**
break a process	**kill** or CTRL-C
copy files	**cp** or **tar**
change virtual screens	ALT-*functionkey*
check disk usage	**df** and **du**
check process status	**ps**
check user status	**finger**, **who**, or **whodo**
communicate with others	**elm** or **talk**
create a directory	**mkdir**
display or type a file	**cat**, **head**, **more**, or **tail**
edit files	**vi**
list files	**ls**
logout	**exit** or CTRL-D
read on-line help	**man**
set your password	**passwd**

UNIX command reference, cont.

Administrative Commands

fsck	examine and repair the hard disk
init	change run level
lpadmin	set up or change a printer
shutdown	shutdown the system
uadmin	reboot or shutdown the system
uname -S	name your system

Network Commands

ftp	start the file transfer program
ping	determine if a system is active
rcp	copy files between systems (remote copy)
rlogin	login to another system (remote login)

Commands Used to Configure Amiga UNIX

Use...	to...
alias	customize command names and shortcuts
chgrp	change the user group a file belongs to
chmod	change file access permission
chown	change file ownership
ln	link a "fake" file to a real file
lpadmin	configure the printers attached to your system
mkfs	create a file system
mount	mount a partition
passwdall	assign a system password
pwconv	hide user password information
set	set and display environment variables
stty	set or display system input/output options
uname	name your system

UNIX command reference

Why read this chapter?

This chapter contains:

- a quick reference chart of commands grouped by function
- brief descriptions (including formats, restrictions, and examples) of commonly used commands
- charts of special characters you can use in commands

Shell differences

The Korn shell (ksh) is the preferred shell because it is an enhanced form of the standard shell (sh). The descriptions in this chapter are based upon how commands work in ksh. The C shell (csh) is not compatible with sh, so some commands work differently in csh, in particular: alias, history, oladduser, set, setenv, and type. Special notes included in the descriptions explain how these commands work differently in C shells.

Command Reference Chart

Key phrase	Related command	Description	Page
active users	who	user status	355
"	who am i	display user information	355
active processes	jobs	list your processes	292
"	ps	list active processes	319
"	whodo	display user process status	356
alter screen	color	change screen color	265
"	set	change terminal variable	333
bring down system	shutdown	shutdown system	337
calculator	bc	binary calculator	255
calendar	cal	on-line calendar	256
cancel	cancel	cancel a print request	257
"	kill	cancel background process	293
change directory	cd	change directory	259
change ownership	chgrp	change group	260
"	chown	change owner	263
change permissions	chmod	change permissions	261
clear screen	clear	clear screen	264
clock	date	display date and time	270
communicate	elm	electronic mail	274
"	mail	electronic mail	302
"	mesg	message setting	304
"	rn	on-line news	328
"	talk	on-screen message system	344
"	wall	send message to all users	354

Command Reference Chart, cont.

Key phrase	Related command	Description	Page
copy	cp	copy file or directory	266
"	cpio	copy to/from device	267
"	rcp	copy to remote system	322
"	tar	copy to/from device	345
copy output to file	tee	copy output to file	347
create directory	mkdir	create directory	305
create file	>	redirect output to a file	362
"	vi	visual editor	353
customize command	alias	customize commands	253
disk	df	display disk allocation	271
"	du	display disk usage	272
"	fdfmt	format disk	279
"	fsck	repair disk problems	284
display file	cat	display file contents	258
"	head	display beginning of file	289
"	less	display file contents	294
"	more	display a page at a time	308
"	pg	display a page at a time	316
"	tail	display end of file	343
display directory	du	display disk usage	272
"	ls	display directory contents	300
"	pwd	display current directory	321
echo	echo	echo output	273
editor	emacs	file editor	276
"	sed	stream editor	331
"	vi	file editor	353

Command Reference Chart, cont.

Key phrase	Related command	Description	Page
file system	mkfs	create file system	306
"	mount	load file system	309
"	rdb	partition hard drive	323
file type	file	display file type	280
find	apropos	find term in **man** pages	254
"	find	find file or directory	281
"	grep	find term in file	288
help	man	on-line reference	303
link	ln	link files or directories	295
move	mv	move/rename file or directory	311
network	ftp	File Transfer Protocol	286
"	ping	check status of system	317
"	rcp	copy to/from another system	322
"	rlogin	log in to another system	325
"	telnet	log in to another system	348
OPEN LOOK	oladduser	set up OPEN LOOK defaults	312
"	olinit	start up OPEN LOOK	313
password	passwd	change password	314
"	passwdall	change or delete all passwords	315
"	pwconv	update /etc/shadow file	320
past processes	acctcom	display past process statistics	252
"	history	display command history	290
pathname	type	display command pathname	350
"	tty	display terminal pathname	349

Command Reference Chart, cont.

Key phrase	Related command	Description	Page
printing	cancel	cancel a print request	257
"	lp	print files	296
"	lpadmin	define print services	297
"	lpstat	display print request status	299
"	pr	format a file	318
remove	rm	remove files	326
"	rmdir	remove directories	327
rename	mv	rename/move file or directory	311
sort	sort	sort lines of screen or file	341
spreadsheet	sc	spreadsheet program	330
start-up	crontab	execute scheduled tasks	269
"	init	initial set-up process	291
"	olinit	initialize OPEN LOOK	313
suspend process	sleep	suspend process	340
terminal information	stty	change/display terminal info	342
"	tty	display terminal pathname	349
xterm	xhost	list systems with access to X	357
"	xset	set user preferences	358
"	xterm	create an Xterm window	360

acctcom — list process statistics

Description

Provides statistics about system usage. Searches the process accounting files and displays records of past activity. **acctcom** reports only on processes that have been terminated.

Restrictions

acctcom only works if you turn on process accounting. To do this, as root, link /etc/init.d/acct to /etc/rc2.d/S30acct; then reboot.

Formats

acctcom — Lists process statistics.

acctcom -b — Lists process statistics backwards in time, showing what just happened.

acctcom -u *username*

Lists statistics for processes executed by the user.

acctcom -n *command*

Lists statistics for all occurrences of the command.

Examples

- Find out what Joe's been working on all day.

```
acctcom -u joe
```

- Find out how many times someone deleted files.

```
acctcom -n rm
```

alias — customize commands

Description

Allows you to define your own commands (aliases) using existing UNIX commands. Typing the **alias** executes the commands you include in the definition.

Restrictions

Available only with the C (csh) or Korn (ksh) shells.

C shell note

The C shell format is alias term command(s).

Formats

alias — Displays the existing **aliases**.

alias *term* — Displays the command assigned to the **alias**.

alias ***term="command(s)"***

Defines a new term for a command or a series of commands.

Examples

- Display the **alias** for **psf**.

```
alias psf
psf=ps -f
```

- Define **psf** as the **alias** for the **ps -ef** (status of all users) command sequence in the K shell.

```
alias psf="ps -ef"
```

- Define **psf** as the **alias** for the **ps -ef** command sequence in the C shell.

```
alias psf ps -ef
```

apropos — search man pages for keyword

Description

Displays the titles of the **man** pages that contain the keyword you specify. You can use this command to determine if there is a **man** page for any keyword or phrase.

Format

apropos *term*	Displays all titles of **man** pages that contain the term.

Example

- Display all **man** pages that include the term "talk" in their title.

```
apropos talk
```

```
talk (1)
talk to another user
```

bc

start binary calculator

Description

Invokes a calculator. Refer to the **bc man** pages for details about advanced options. The default mode for division is integer only.

Format

bc Displays a blank line to use for calculations.

Examples

- Start the calculator.

  bc

- Perform calculations.

  243 + 100 and press RETURN

  ```
  343
  ```

  25-5 and press RETURN

  ```
  20
  ```

  24/6 and press RETURN

  ```
  4
  ```

  12*4 and press RETURN

  ```
  48
  ```

- Check the current number of places past the decimal point.

  scale

  ```
  0
  ```

- Change the current number of places past the decimal point to 2.

  scale=2

cal — display calendar

Description

Displays a calendar.

Formats

cal	Displays a calendar for current month.
cal ***year***	Displays a calendar for specified year.
cal ***month year***	Displays a calendar for specified month and year.

Restrictions

You must enter the full year; if you enter 91 rather than 1991, a calendar for the year 91 (rather than the year 1991) displays.

Examples

- Show a calendar for this month (October, 1990).

```
cal
```

```
   October 1990
 S     M  Tu   W  Th   F   S
       1   2   3   4   5   6
 7     8   9  10  11  12  13
14    15  16  17  18  19  20
21    22  23  24  25  26  27
28    29  30  31
```

- Show a calendar for December of 1991.

```
cal 12 1991
```

```
   December 1991
 S     M  Tu   W  Th   F   S
 1     2   3   4   5   6   7
 8     9  10  11  12  13  14
15    16  17  18  19  20  21
22    23  24  25  26  27  28
29    30  31
```

cancel

cancel print job

Description

Cancels or stops a request sent to the printer. When you issue an **lp** command to print a file, **lp** assigns a request ID to the file. To stop printing your file, use the **cancel** command and the request ID associated with that file.

Formats

cancel ***request-ID***

Cancels a specific print job.

cancel -u ***username***

Cancels all jobs requested by this user.

cancel ***printername***

Cancels the current job on this printer.

Example

- Cancel a job you started. First, find the request ID. Then, cancel the job, using the same request ID.

```
lpstat
ps_tty06-587    joe     134    October 19 11:32 on pr1
cancel ps_tty06-587
request "ps_tty06-588" cancelled
```

cat — display a file

Description

Displays the contents of a file.

If a file has more than one screen of information to display, the screen scrolls to the end of the file. If you want to see one page at a time, use another display command (**more**, **less**, **pg**).

Formats

cat ***filename***	Displays the contents of the file.
cat	Takes input from the keyboard instead of a file; used with redirection to create (>) or add to (>>) a file.

Examples

- Display the contents of the /etc/profile file.

```
cat /etc/profile
```

```
# ident "@(#)sadmin:etc/profile 2.3"
#  The profile that all logins get before using their own
                    >    (screen scrolls to end)
umask 022
```

- Create a file (file3) that contains file1 and file2.

```
cat file1 file2 > file3
```

```
This is the text for file1.
This is the text for file2.
```
The two files displayed through **cat** become file3.

- Create the temp file from keyboard input; end input with CTRL-D.

```
cat > temp
This is a test line. <CTRL-D>
```

cd — change directory

Description

Changes the current directory.

Formats

cd	Changes to your home directory.
cd /	Changes to the root directory.
cd ..	Changes to the directory above (the parent directory).
cd ***directory***	Changes to the specified directory.

Examples

- Change from any directory to Joe's home directory.

```
cd
```

- Change from /home/joe to /home/mary.

```
cd ../mary
```

- Change from any directory to /usr/public/src.

```
cd /usr/public/src
```

- Move up to the public directory, which is one level above the current directory.

```
cd ..
```

chgrp — change group

Description

Changes the group that a file belongs to. You can specify any group name listed in the group ID file (/etc/group).

The group associated with the file may have special read, write, and execute permissions that allow members to access the file while preventing users from other groups from accessing it.

Restrictions

You must be the owner of the file or root to use this command.

Formats

chgrp *group file* Changes the group of a file.

chgrp -R *group directory*

Changes the group of a directory and all the files in the directory.

Examples

- Give the lab group permission to use the chem file.

```
chgrp lab chem
```

- Give the lab group unlimited access to all files in the testing directory.

```
chgrp -R lab testing
```

chmod — change permissions

Description

Changes the access mode of a file. The **chmod** command sets a file or directory's read, write, and execute permissions for the owner, the group, and other users.

Restrictions

You must be root or the owner of the file or directory to use this command.

Formats

chmod ***user action permission file***

Change the file permissions for all types of users.

chmod -R ***user action permission directory***

Make **chmod** recursive (affect the directory and all files and directories below it in the directory structure).

The *user*, *action*, and *permission* options are listed together without spaces; they're treated as one argument. The table on the next page lists the user, action, and permission options.

You can use any combination of options as long as you include a user (owner, group, other, all users), an action (add, remove, set), a permission (read, write, execute), and a file or directory name.

chmod, cont.

Options

Option	Description
User	
u	owner of the file
g	group to which the owner belongs
o	all other users
a	all users; can be used to represent the combination of **ugo**
Action	
+	add permission
-	remove permission
=	set privileges exactly
Permission	
r	read permission; able to read a file, or list a directory's contents
w	write permission; able to make changes to a file; or delete, copy, and rename files within a directory
x	execute permission; able to execute a file or access a directory

Examples

- Remove write (w) permission from the group (g) for the animal file.

```
chmod g-w animal
```

- Allow all users (a) to read (r), write (w), and execute (x) the office file.

```
chmod a+rwx office
```

chown — change owner

Description

Changes the owner of a file or directory.

Restrictions

Only root or the owner of a file can change the owner of a file or directory.

Formats

chown ***owner file***

Changes the owner of the file.

chown ***owner directory***

Changes the owner of a directory.

chown -R ***owner directory***

Changes the owner of the directory, including all files and subdirectories.

Examples

- Give Joe ownership of all the files in the product directory.

```
chown -R joe product
```

- Make Jane the owner of the absence log.

```
chown jane absence.log
```

clear — clear screen

Description

Clears your current screen.

Format

clear Clears the current screen.

color — change screen colors

Description

Changes the screen colors.

You can change the background and foreground colors on a screen by specifying either the name of a color or a value that represents the color.

The color names you can use are black, blue, cyan, green, magenta, red, white, and yellow.

Numeric values for colors are between 000 and fff; each position in the value represents an amount of red, green, and blue, respectively. Changing these numbers individually allows 4096 color combinations between the darkest, 000, and the lightest, fff.

Formats

color -show

Lists the current screen colors. The first two listed (c00 and c01) are the current background and foreground colors, respectively.

color -bc *color* -fc *color*

Changes the background and foreground colors of a screen.

Example

- Change the background color to black and the foreground color (letters) to red.

```
color -bc black -fc red
```

cp — copy files

Description

Copies one file to another file on the hard disk.

Formats

cp *file newfile* Copies the file to the new file.

cp *directory/file newdirectory*

Copies the file to the new directory.

cp -r *directory newdirectory*

Copies an entire directory tree to a new directory. The **-r** option makes **cp** recursive, so it finds and copies all subdirectories.

Examples

- Make a copy of the dogs file and call the new file pets.

```
cp dogs pets
```

- Make a copy of the /usr/public/office directory and all its files and subdirectories. Call the new parent directory/usr/public/admin.

```
cp -r /usr/public/office
                /usr/public/admin
```

- Copy all files in the current directory to /tmp.

```
cp * /tmp
```

cpio — copy in and out

Description

Copies files to or from a hard disk, floppy disk, or tape.

Before using **cpio** to write or copy files, use a file list command (**ls**, **find**) as input to **cpio**; this provides accurate pathnames to the files you want to write or copy. **cpio** creates an archive file which can only be restored with a **cpio** command.

Note

cpio has many options. The combinations listed here are fairly standard for output (-ocv -O), input (-imdcuv), and list (-ictv). In addition, you should include the -B option when writing to or restoring from a tape.

Restrictions

To copy a file or directory to a floppy disk, you must use a formatted disk. The **cpio** command does not format a disk. To format a floppy disk, refer to the **fdfmt** command summary.

Formats

file list **| cpio -ocv >** ***filename***

Writes all files listed by a file list command (**ls**, **find**) to the specified file.

file list **| cpio -ocv -O >** ***device***

Writes all files listed by a file list command to the specified device, prompting for new disks or tapes as each one fills up.

cpio, cont.

cpio -imdcuv ***filename < device***

Retrieves only the specified file from the **cpio** archive on the device.

cpio -imdcuv < ***device***

Retrieves all files from the specified device.

cpio -ictv < ***device***

Lists the files contained on the device.

file list **| cpio -pdmv** ***directory***

Copies all files listed by the file list command to the directory. These files are not copied in **cpio** format, so they can be accessed using other commands.

Examples

- Copy the contents of your home directory to floppy disks in the internal drive.

```
ls /home/joe | cpio -ocv -O >
               /dev/dsk/fd0
```

- Retrieve a **cpio** backup from tape.

```
cpio -imdcuv -B < /dev/rmt4
```

NOTE: This command replaces existing files on your hard disk.

crontab

create cron table

Description

Creates a file of commands you want executed by the **cron** process. **cron** is an automatic process that reads and executes files in the /var/spool/cron/crontabs directory. These files include scheduling information and commands for tasks that run on a regular schedule (backups, deletes, listings, messages).

The entries in these files consist of six fields, each separated by spaces or tabs. The first five fields define when to execute the command in the sixth field. The first five fields specify the minute (0 to 59), the hour (0 to 23), the day of the month (1 to 31), the month (1 to 12), or the day of the week (0 to 6, where 0 is Sunday).

Restrictions

You must be root and add a username after the option to change, remove or list any other user's crontab file.

Formats

crontab -e Edit your crontab file, or create an empty file to edit if **crontab** doesn't exist.

crontab -l List your **crontab**.

crontab -r Remove your **crontab**.

Example

- List Joe's crontab file.

```
crontab -l joe
```

```
00 17 * * 1,2,3,4,5 /usr/bin/ckbupscd>/dev/console
```

This crontab entry runs a check for backup program on the console screen (F1) every Monday through Friday at 5pm.

date — set or display date and time

Description

Displays or sets the date and time.

Restrictions

Only root can set the date. However, any user can display it.

Formats

date — Displays current date and time.

date *mmddHHMMyy*

Sets the current date; *mm*, *dd*, *HH*, *MM*, and *yy* represent the current month, date, hour, minutes, and year respectively. Use military hours for the time (00-24).

date *HHMM* — Change only the time.

Examples

- Display the time and date.

```
date
```

```
Fri Oct 19  13:42:13  EDT  1990
```

- Set the system clock to January 8, 1991 1PM.

```
date 0108130091
```

```
Tue Jan 8  13:00:00  EST  1991
```

df — calculate free disk space

Description

Displays the disk space used and available for each mounted file system.

Formats

df — Displays free disk space in blocks (512 bytes per block).

df -k — Displays disk usage in kilobytes, including total space, space used, space available, and percentage used.

Example

- Find out how much space is left on your hard disk (/dev/dsk/c0d0s1).

```
df -k
```

```
filesystem      kbytes    used   avail  capacity mounted on
/dev/dsk/c0d0s1 79872   68682   11190     86%    /
proc                0       0       0       0    /proc
fd                  0       0       0       0    /dev/fd
```

du — display disk usage

Description

Displays the number of blocks (512 bytes) contained in each file or directory.

Formats

du	Displays the number of blocks for each directory.
du -a	Displays the number of blocks for each directory and file.
du -s	Displays the grand total of blocks used by the current directory.

Examples

- Find out how much space your current directory is taking up and list the size of each file.

  du –a

```
2       ./.profile
12      ./.sh_history
2       ./.Xdefaults
        >
2       ./temp
1136    .
```

- Find out the total amount of blocks used by your current directory.

  du –s

```
13304   /usr/bin
```

echo

echo output

Description

Displays the results of a command, the value of a variable, or the text you typed on the terminal screen. You can use **echo** to display program status while debugging a command file.

Formats

echo ***phrase***	Displays the phrase.
echo $*variable*	Displays the value of the shell environment variable.

Examples

- Display the phrase "TIME TO GO" in one hour.

```
sleep 3600;echo TIME TO GO
```

```
TIME TO GO
```

- Display the pathname of your home directory.

```
echo $HOME
```

```
/home/joe
```

elm — start electronic mail program

Description

Invokes **elm**, an electronic mail program. For more **elm** information, refer to the *Using electronic mail (elm)* chapter in this manual.

Formats

elm

Allows you to read, send, delete, and store mail messages from the **elm** mailbox screen.

elm ***username***

Sends mail without going through the **elm** mailbox screen.

elm -s ***subject username < filename***

Sends a file to another user.

Examples

- Send a message to Joe without going through the **elm** mailbox screen.

```
elm joe
```

Your editor opens a file for your message; elm sends the message when you confirm the send option.

- Send the office procedures file to Joe.

```
elm  -s  procedures  joe < office.proc
```

```
Sending mail...
Mail sent!
```

Mailbox screen options

These options are available on the mailbox screen.

Options	Key	Description
alias	a	change to alias mode
change	c	change to another folder
copy	C	copy current or tagged messages to a folder
delete	d	delete a message
down	j	move down the mail index
forward	f	forward a message to another user
group reply	g	reply to everyone who received the current message
up	k	move up the mail index
mail	m	mail a message
options	o	access **elm** options
print	p	print a message
quit	q	exit from **elm**
reply	r	answer a message
save	s	save a message in a folder
undelete	u	cancel the delete request
quick quit	x	quit without changing
search	//	search for item in message
help	?	find out more information about a specific command
next	+	display the next ten message lines
previous	-	display previous ten message lines
read	RETURN or SPACE	read a message
redraw	CTRL-L	redraw a screen

emacs

screen editor

Description

Invokes the **emacs** editor, an editor used as an alternative to the **vi** editor. **emacs** allows you to:

- edit several files simultaneously
- open multiple windows on the same document
- define keyboard macros
- customize its commands to fit your needs
- create your own commands by using the Lisp programming language

emacs is distributed under a general public license and is not part of the Amiga UNIX system software. More information is in the *GNU Emacs Manual* published by the Free Software Foundation (675 Massachusetts Avenue, Cambridge MA 02139).

emacs commands

Keys	What happens?
emacs	start **emacs**
CTRL-X then CTRL-C	exit **emacs**
CTRL-Z	suspend **emacs**
CTRL-X then CTRL-F *file*	create a file
CTRL-X then CTRL-F *file*	read an existing file
CTRL-X then CTRL-S	save a file
CTRL-X then CTRL-D	list files
CTRL-^	get help
CTRL-^ T	start the tutorial

env

display environment variables

Description

Displays environment variables. Environment variables are shell variables that are passed to programs like **mail**, **elm**, and **vi** for use in their processing.

Note

env is closely related to the **setenv** and **set** commands; read those *Command Reference* pages for more information.

Formats

env — Displays the values of all environment variables (those variables used by commands invoked from your current shell).

Example

- List your environment variables.

```
env
```

```
_=/usr/bin/env
AZ=100
PATH=/home/carol/bin:/usr/local/bin:/usr/
                    ^
PWD=/home/carol
TZ=EST5EDT
```

exit — log out of shell

Description

Logs you out of your current shell.

If your current shell is the only shell, **exit** logs you out of the system. Typing **exit** at the shell prompt is the same as pressing CTRL-D.

fdfmt — format floppy disk

Description

Formats a floppy disk in the floppy disk drive.

Format

fdfmt > *device* Formats a disk in the device.

Examples

- Format a disk in the internal floppy disk drive.

```
fdfmt > /dev/rdsk/fd0f
```

- Format a disk in a second internal floppy disk drive connected to your system.

```
fdfmt > /dev/rdsk/fd1f
```

- Format a disk in an external floppy disk drive.

```
fdfmt > /dev/rdsk/fd2f
```

file — show file type

Description

Determines and displays the file type (for example, directory, executable, commands, or English text file).

Formats

file ***filename***	Determines and displays the file type.
file *	Determines and displays the file type of each file in your current directory.

Examples

- Find out the file type of the file named pets.

```
file pets
```

```
pets:                   English text
```

- Find out what types of files are in your directory.

```
file *
```

```
Cancelled mail:         empty file
Mail:                   directory
News:                   directory

time.clock:             commands text
trash:                  ascii text
```

find — find files

Description

Searches a directory tree to locate a file.

Formats

find ***directory*** **-name** ***filename*** **-print**

Find the file in the directory tree and display its name on the screen. (The **-print** option prints the name on your screen.)

Examples

- Find and display the pathnames of all files named pets in the current directory and its subdirectories.

```
find . -name pets -print
./pets
```

- Find and display the pathnames of all files whose names begin with "m" in the zoo directory and its subdirectories.

```
find zoo -name "m*" -print
./messages
./memos
./mystuff
```

finger — display user information

Description

Displays status information about each user currently logged in to your system, including:

- username
- full name
- virtual screen (tty and console)
- idle time
- time when user logged in
- location (if known)

Formats

finger — Displays the status of users who are logged in to the system.

finger -l — Displays user status in more detail.

finger *username* — Displays detailed status about the specified user.

finger *username@system* — Displays the status of the user on the specified system on a network.

Examples

- Find out who is logged in to your system.

```
finger
```

```
Login    Name            TTY         Idle    When      Where
root     The Super User  console     1:41    Fri 12:55
joe      Joe Smith       term/con2           Fri 09:50
```

Finger

display user information

Description

Finger (with a capital "F") is a special public domain program available with Amiga UNIX. Like **finger** (with a lower case "f"), this command displays information about each active user on the system; in addition, **Finger** supplies information about the processes and shells each user is running.

Formats

Finger	Displays information about active users and their processes and shells.
Finger -l	Displays additional information about each active user, including: home directory, user number, group number, and shell.
Finger *username*	Displays detailed status about the specified user.
Finger *username@system*	Displays the status of the user on the specified system.

Example

- Find out what programs are running on each virtual screen in the utopia system.

Finger

```
[utopia]
NAME     GROUP    FULLNAME      WHAT  IDLE   TTY    LOCATION
carol      doc    Carol Jones   ksh     41   pts/5  writers1
joe      sales    Joe Smith     ksh     30   pts/2  narnia
robert     mkt    Robert Green  csh      6   pts/3  carol
```

fsck — file system check

Description

Examines your hard disk and makes repairs if necessary. This process runs automatically when UNIX is not shut down properly and tries to enter multi-user mode during reboot.

fsck (without options) asks you to confirm any disk repair that needs to be made. If **fsck** cannot determine a file's name, it places the file into the lost+found directory in the top level of the file system being modified.

WARNING: Never use **fsck** on a mounted file system. If you need to check or repair your hard disk, change to single-user mode first.

Formats

fsck
Examines the file system; prompts the user to confirm repairs.

fsck -m
Examines but doesn't repair the file system; used to verify that a file system is suitable for mounting.

fsck -y
Assumes a yes answer to all questions and automatically repairs any problems. This option is used during reboot.

fsck, cont.

Example

- Check your external hard drive for problems.

```
fsck /dev/dsk/c3d0s1
```

```
/dev/dsk/c3d0s1
File System:          Volume:

**  Phase 1 - Check Blocks and Sizes
    Phase 2 - Check Pathnames
    Phase 3 - Check Connectivity
    Phase 4 - Check Reference Counts
    Phase 5   Check Free List
 117 files  20898 blocks  126756 free
```

ftp — Network File Transfer Protocol

Description

The File Transfer Protocol (**ftp**) allows you to access another system on a connected network and perform several file and directory functions, including copying and deleting.

After you connect to the other system (*remotesystem*), **ftp** asks you to log in. After you log in, the **ftp** prompt (ftp>) appears. You can use **ftp** commands to perform file and directory tasks. A few of the commands are listed below; refer to the **ftp man** pages for a more complete list.

Format

ftp ***remotesystem***

Access the remote system.

ftp commands

Once you have logged in to the remote system, you can use the **ftp** commands. In the following examples, "local" refers to files and directories on your original system; "remote" refers to files and directories on the system with the **ftp** prompt.

bye — Break the session with the remote system.

delete ***remotefile***

Delete a file on the remote system.

get ***remotefile localfile***

Move the remote system's file to your local system.

help — List the **ftp** commands.

help ***command*** Display a description of the command.

put ***localfile remotefile***

Move the local file onto the remote system. If you do not provide a remote filename, **put** gives the remote file the same name as the local file.

Examples

- Access the abc system.

```
ftp abc
```

```
Connected to abc
220 abc FTP server (UNIX(r) System V Release 4.0) ready.
Name (abc:carol): <username>
331 Password required for carol.
Password:  <password>
730 User carol logged in.
ftp>
```

- Copy the library file from the abc system to your local system. Give the new file the same name as the remote filename.

```
get library
```

```
200 PORT command successful.
150 ASCII data connection for file 162.6.201.143,1051)(307)
226 ASCII Transfer complete.
local:library  remote:library
321 bytes received in 0.017 seconds (19 Kbytes/s)
ftp>
```

grep — search files for a phrase

Description

Searches files for a specified word or phrase and displays all the lines that contain the word or phrase.

Formats

grep *phrase file(s)*

Searches the file(s) for the word or phrase and displays any line containing it.

grep -i *phrase file(s)*

Searches files for the phrase and ignores letter case.

grep -n *phrase file(s)*

Searches files for the phrase and lists the line number the phrase is found on.

Restrictions

If the phrase has more than one word, close it in single quotes.

Examples

- Search for the word "home" or "HOME" in the .profile file.

```
grep -i home .profile
```

```
. SHOME/.olsetup        #!@Do not edit this line !@
```

- See if the phrase "do not" is in any file in the current directory.

```
grep 'do not' *
```

```
ftp:    -n   Do not attempt auto-login upon initial
```

head

display first part of a file

Description

Displays the first 10 lines of a file. You can also specify the number of lines displayed.

Formats

head ***file***	Displays the first ten lines of a file.
head -*n file*	Displays the first *n* lines of the file.

Examples

- Display the first ten lines of the office.proc file.

```
head office.proc
```

- Display the first 25 lines of the zoo file.

```
head -25 zoo
```

history — display previous commands

Description

Displays a list of commands you recently typed at the shell prompt.

C shell note

This command works in the Korn and C shells. However, for **history** to be recorded in the C shell, you must set the history variable to a number of lines (**set history =** ***n***).

Formats

history	Displays previously executed commands; **history** is the last command displayed.
history -r	Displays previously executed commands in reverse order; **history -r** is the first command in the list.

Examples

- List the commands you most recently typed during this session.

```
history
```

```
477 finger
478 finger joe@utopia
479 finger @utopia
    >
492 history
```

- Find out whether you ordered a printout of a specific file by looking for the **lp** command in the command history.

```
history | grep lp
```

```
481 lp -o postscript test
496 history | grep lp
```

init — initialize system processes

Description

Controls processes for virtual screens, ports, and init levels from information stored in the /etc/inittab file.

init scans and executes commands specified in /etc/inittab.

Restrictions

You must be root to execute the **init** command.

Format

init ***runlevel*** Reads /etc/inittab and executes any commands associated with the run level.

Run levels

Run level	Description
0	shuts down the system safely
1	system administrator mode
2	multi-user mode
3	remote file sharing mode
S or s	single-user mode
6	reboot

Examples

- Reboot the system.

```
init 6
```

```
INIT:  New run level 6
System coming down.  Please wait.
System services are now being stopped.
```

(The system reboots and displays a login prompt)

- Shutdown the system.

```
init 0
```

```
INIT:  New run level 6
System coming down.  Please wait.
System services are now being stopped.
The system is down.
```

(Turn the machine off, then on, to reboot.)

jobs — list background jobs

Description

Lists your background processes (jobs).

Restrictions

This command works only in the C and Korn shells.

Formats

jobs	Displays your background processes.
jobs -l	Displays your background jobs in a long format.

Examples

- Find out what background jobs are running at the present moment.

```
jobs
```

```
[1]  + Running                 sleep 100 &
```

- List all your background jobs and their PIDs.

```
jobs -l
```

```
[1]  + 177  Running                 sleep 100 &
```

kill — terminate a process

Description

Terminates a process.

Restrictions

You must log in as root or own the process to kill it.

Formats

kill ***PID*** Kills the process specified by the PID.

kill **%** ***jobnumber*** Kills the process specified by the job number.

Examples

- Stop a background process called **sleep**. First, find the PID for **sleep**, and then kill the process using the PID.

```
ps
```

```
PID  TTY          TIME   COMD
139    term/con2   0:02   ksh
177    term/con2   0:00   sleep
178    term/con2   0:00   ps
```

```
kill 177
```

```
[1] + Terminated                    sleep 100 &
```

- Stop a background job. First, list the jobs, and then kill the sleep 100 & job.

```
jobs
```

```
[1] + Running                       sleep 100 &
```

```
kill %1
```

```
[1] + Terminated                    sleep 100 &
```

less — display a file

Description

Displays the contents of a file.

Format

less ***filename*** Displays the contents of a file.

Movement keys

Once the file is displayed, use the following keys to move backward and forward through a file.

Key(s)	Function
SPACEBAR	move forward one screen
RETURN	move forward one line
b	move back one screen
H	help
q	quit and return to the shell prompt
/term	search for *term*
!*command*	invoke a shell command

Example

- Display the /etc/inittab file.

```
less /etc/inittab
```

ln — link files or directories

Description

Creates either a hard or symbolic link.

A hard link creates a name that points to another file; it cannot point to directories or across file systems.

A symbolic link creates a name that points to the original directory or file; it can point across file systems.

A link allows a single file or directory with two (or more) names.

Formats

ln ***file newfile*** Hard links a file to a new filename.

ln -s ***directory newdirectory***

Symbolically links a directory to a new name.

Examples

- Create a jobs file that points to the tools file.

```
ln tools jobs
```

- Create and symbolically link the /usr/games directory with Joe's home directory. List the new directory.

```
ln -s /usr/games /home/joe/games
ll /home/joe
```

```
total 2
drwx------2 carol other 32  Oct 4  10:07 Mail
-rw-r--r--1 carol other 17  Oct 18 14:51 games->/usr/games
```

lp — print files

Description

Prints a file or several files. Unless you specify a particular printer, **lp** uses the default printer.

Formats

lp ***file***	Prints the file.
lp ***file file***	Prints multiple files.
lp < ***file***	Prints a file which does not have read permission set for other users.
lp -n ***n file***	Prints *n* copies of the file.
lp -d ***printer file***	Prints the file on the specified printer.
lp -o ***options file***	Prints the file using options for your printer interface.

Examples

- Print the pets file.

```
lp pets
```

- Print the PostScript test file on your PostScript printer.

```
lp -o postscript test.file
request id is post2-17 (1 file)
```

- Print the sales file on the printer named billing, without a banner page.

```
lp -d billing -o nobanner sales.file
request id is post2-17 (1 file)
```

lpadmin — administer printers

Description

This command performs several print service functions:

- defines printers and devices
- adds, changes, and removes printers
- sets or changes the system defaults
- defines user alerts for printer defaults

There are many configuration options with the **lpadmin** command. However, only four options are mandatory to define a printer connected to your Amiga: default (-d), printer name (-p), device (-v), and interface (-i).

Note

To setup printing, you need to use the **enable** and **accept** commands. See the *Printing* chapter for instructions.

Restrictions

You must be root to use this command.

Formats

lpadmin -p ***name*** **-v** ***device*** **-i** ***interface***

Defines a printer (*name*) for your system.

lpadmin -d ***name***

Defines the default printer.

lpadmin, cont.

Examples

- Define a printer named p1 as a PostScript printer on the main serial port.

```
lpadmin -p p1 -v /dev/term/ser -i
    /usr/spool/lp/model/postscript
```

- Define a PostScript printer on the parallel port.

```
lpadmin -p p2 -v /dev/par -i
    /usr/spool/lp/model/postscript
```

- Make p1 the default printer.

```
lpadmin -d p1
```

lpstat

report printer status

Description

Displays the status of any requests sent to the printer and the status of the printer. You can use the request ID displayed by the **lpstat** command with the **cancel** command to stop printing.

Formats

lpstat — Displays the status of requests you sent to the printer.

lpstat -d — Identifies the default printer.

lpstat -p *printername*

Lists requests sent to the specified printer.

lpstat -u *username*

Lists requests sent to the printer by the user.

lpstat -o — Displays the printer status and all print requests.

lpstat -v — Displays the device names for the printers on your system.

Example

- Check the status of all print requests.

```
lpstat -o
```

```
post2-17          carol          17     Oct 20 15:01
```

ls — list file and directory information

Description

Lists information about the files and subdirectories contained in a directory.

Formats

ls *-option*
Lists the files contained in your current directory. The format of the list changes with the options. Without an option, **ls** displays an alphabetical list.

ll
Predefined shortcut for **ls -l** (long format list).

lf
Predefined shortcut for **ls -F** (displays special characters indicating directories (/) and executable programs (*)).

Options

Option	Description
a	display all files, including files that begin with a period
F	display special characters to indicate directories and executable programs
l	long format
r	reverse the sort order
t	display files by time stamp (latest first)

ls, cont.

Examples

- List the contents of Joe's home directory in the long format.

```
ll /home/joe
```

```
drwx------  2   joe mgt 32   Oct  4  10:07 Mail
lrwxrwxrwx  1   joe mgt 10   Oct 20  14:55 games->/usr/games
drwxr-xr-x  2   joe mgt 32   Oct 18  12:34 memos
-rw-r--r--  1   joe mgt 17   Oct 10  14:3  meeting1
```

- List your files by the time they were last modified, ending with the files you most recently changed (reverse time order).

```
ls -rtl
```

```
drwx------  2   joe mgt 32   Oct  4  10:07 Mail
-rw-r--r--  1   joe mgt 17   Oct 10  14:30 meeting1
drwxr-xr-x  2   joe mgt 32   Oct 18  12:34 memos
lrwxrwxrwx  1   joe mgt 10   Oct 20  14:55 games->/usr/games
```

- Find out if .profile is in your home directory. (You're currently working in your home directory.) List all filenames, including those starting with a period.

```
ls -a
```

```
.               .elm            .profile    .signature  games
..              .oldinitrc      .rhosts     .signature2 junk
.Xdefaults      .olsetup        .sh_history Mail        memos
```

mail — read and send electronic mail

Description

Invokes an electronic mail program. This program provides some of the same functions as **elm**, another electronic mail program, except **mail** doesn't display a list of messages or a menu of options.

Formats

mail — Displays mail sent to you by other users.

mail ***user*** — Sends mail to a user. End the message with a period on a blank line.

mail ***user@system***

Sends mail to a user at the specified system.

Options

Option	Description
?	help
RETURN	display next message
d	delete message
p	print message
q	quit
r	reply to message
s	save message in your mailbox

Examples

- When you log in to your system, a notice tells you that you have mail. Read your mail messages.

```
mail
```

- Send a mail message to Joe.

```
mail joe
```

man — manual pages

Description

Displays the on-line manual pages for a specified command.

Note

Some commands have more than one set of man pages. For example, **echo** has a page for AT&T/Amiga UNIX, a page for Berkeley UNIX, and a page for a different FMLIU (Form and Menu Language Interpreter Utilities) command. All three **man** pages are displayed in sequence; the first one is probably the most useful.

Format

man ***command*** Displays the **man** page for the specified command.

Example

- Find out about options that are available with the **lpstat** command.

```
man lpstat
```

```
lpstat(1)                    USER XCOMMANDS                    lpstat(1)

NAME
     lpstat - print information about the status of the LP
     print service

SYNOPSIS
     lpstat [options]

DESCRIPTION
     The lpstat command prints information about the current
     status of the LP print service.

     If no options are given, then lpstat prints the status
```

mesg — allow or deny messages

Description

Allows you to decide whether your screen can be interrupted by messages from other users.

Formats

mesg	Displays your message receiving status (y/n).
mesg y	Allows your terminal to receive messages. (This is the status when you log in.)
mesg n	Prevents your terminal from receiving messages.

Examples

- Find out if you can receive messages.

```
mesg
is y
```

- You don't want to be interrupted by messages from other users.

```
mesg n
```

- You want to accept messages.

```
mesg y
```

mkdir — make directory

Description

Creates a directory.

Formats

mkdir ***directory*** Creates a directory.

mkdir ***directory directory***

Creates several directories at a time.

mkdir -p ***directory***

Creates a directory and all the directories necessary to complete the path.

Examples

- Create a directory named projectx.

```
mkdir projectx
```

- Create three new directories.

```
mkdir procs bills accounts
```

- Create a zoo directory with a pets subdirectory.

```
mkdir -p zoo/pets
```

mkfs — make file system

Description

Creates an empty file system on a hard disk partition, floppy disk, or tape. The file system can be mounted once you create it with **mkfs**.

When you use the **mkfs** command, specify a device and the size of the file system (in file system blocks).

Note

The **mkfs** command waits 10 seconds before starting. During this time, you can abort **mkfs** by pressing the DEL key or CTRL-C.

Format

mkfs -F ***FStype device blocks***

Creates a file system (FS) on the device with the specified number of blocks.

A hard disk uses /dev/dsk/c*x*d0s*y* as the device format (*x* is the SCSI device number and *y* is the partition number). You can change the numbers *x* and *y* to identify the specific SCSI device and partition for each file system (do not use 0 for *y* because 0 identifies the entire hard disk, not a partition).

The floppy disk drive uses /dev/dsk/fd*n* as the device format (*n* is the number of the drive). The standard internal floppy disk is /dev/dsk/fd0 and has 1760 blocks.

mkfs, cont.

Examples

- Make a System V file system with a block size of 76800 (35 Mb) on hard disk /dev/dsk/c1d0s1 (SCSI disk 1).

```
mkfs -F s5 /dev/dsk/c1d0s1 76800
```

- Make a System V file system with a block size of 1760 on the floppy disk in the internal drive (0).

```
mkfs -F s5 /dev/dsk/fd0 1760
```

more — display file contents

Description

Displays the contents of a file or several files, one screen at a time.

Formats

more ***file*** — Displays the contents of a file.

more ***file file*** — Displays the contents of a file followed by the contents of another file.

Movement keys

Once the file is displayed, use the following keys to move backward and forward through a file.

Key(s)	Function
SPACEBAR	move forward one screen
RETURN	move forward one line
b	move back one screen
q	quit and return to the shell prompt
/term	search for *term*
!*command*	invoke a shell command
?	help

Examples

- Display the pets file, then display the dogs file.

```
more pets dogs
```

- Display a yearly calendar, one screen at a time.

```
cal 1991 | more
```

mount

mount file systems

Description

Mounts file systems. A file system must already exist on the device before you can mount it.

When you use the **mount** command, you specify a device and a directory. The directory is the mount point you use to access the mounted file system.

Restrictions

You must be root to mount a file system.

Don't mount a file system to a directory that has data in it. You won't be able to access the original contents of that directory until you unmount (**umount**) the new file system.

Formats

mount — Displays information about file systems already mounted.

mount ***directory*** — Checks the /etc/vfstab file and mounts the file system listed for the specified directory.

mount -F ***FStype device directory***

Mounts the file system (FS) from the device to the directory on the hard disk.

A hard disk uses /dev/dsk/cxd0sy as the device format (*x* is the SCSI device number and *y* is the partition number). You can change the numbers *x* and *y* to identify the specific SCSI device and partition for each file system (do not use 0 for *y* because 0 identifies the entire hard disk, not a partition).

mount, cont.

The floppy disk drive uses /dev/dsk/fd*n* as the device name (*n* is the number of the drive).

Example

- Mount the System V file system on /dev/dsk/c1d0s1 at directory /mnt/joe.

```
mount -F s5 /dev/dsk/c1d0s1 /mnt/joe
```

mv — move or rename files

Description

Moves a file to another directory or renames a file. After a move or rename operation is completed, the source file or directory no longer exists.

Formats

mv *file newfile* Renames the current file with the new filename.

mv *directory newdirectory*

Renames the current directory with the new directory name.

mv *file newdirectory*

Moves a file to a new directory. The file retains its original name.

Examples

- Change the name of the dogs file to pets.

```
mv dogs pets
```

- Rename the directory /home/joe to /home/nina.

```
mv /home/joe /home/nina
```

- Move the report file into the /home/nina directory, keeping the original name.

```
mv report /home/nina
```

oladduser — set up OPEN LOOK defaults

Description

Creates your OPEN LOOK environment so you can use OPEN LOOK's files and default values. This command puts various OPEN LOOK and X Window System resource files in your home directory and modifies your .profile.

oladduser copies the following files from /usr/X/adm into your home directory:

- .Xdefaults
- .olinitrc
- .olprograms

Copies the file .olsetup and adds this line to the .profile file:

- . $HOME/olsetup # do not change line!

C shell note

For C shell users, **oladduser** adds the following lines to the .login file:

- setenv DISPLAY unix:0
- setenv XNETACCES on
- set path=($path /usr/X/bin)

Format

/usr/X/adm/oladduser ***username***

Sets up an OPEN LOOK environment.

Example

- Set up an OPEN LOOK environment for Joe.

```
/usr/X/adm/oladduser joe
```

olinit — start OPEN LOOK

Description

Starts OPEN LOOK, a graphical user interface based on the X Window System, from which you can run terminal windows and programs.

Formats

olinit
Starts OPEN LOOK on your system.

olinit -- bc
Starts OPEN LOOK with the backward compatibility option of X. Use this format if you want to use an application that was written for previous versions of X.

olinit -- *argument*
Passes the argument to the X server; can be any valid X Window System argument.

Example

- Run an application written for version 3 of X.

```
olinit -- bc
```

passwd — set or change password

Description

Sets or changes login passwords.

This command prompts you for your old password, and then prompts you twice for your new password.

Restrictions

If you are logged in as root, you can create and change any user's password by adding the *username* after the command.

A password must be at least six characters long and must contain at least two alphabetic characters and at least one numeric or special character. For example, **test**, **test1**, and **123456** are not valid passwords; **testing1** is.

Format

passwd Allows you to add or change your password.

Examples

- Change your password from billing1 to sales1.

  passwd

  ```
  passwd:   Changing password for joe
  Old password: <billing1>
  New password: <sales1>
  Re-enter new password: <sales1>
  ```

  Type the old password (billing1), and then type the new password (sales1). Verify the new password by typing it again.

- As root, change Joe's password.

  passwd joe

  Type the new password, then confirm it by typing it again. (root does not have to enter the old password.)

passwdall — set or delete system passwords

Description

Sets or deletes passwords for all system accounts.

Restrictions

You must be root to use this command.

Note

This command is specific to Amiga UNIX.

Formats

passwdall — Allows you to enter a standard password for all the system accounts.

passwdall -d — Removes passwords from all system accounts.

Examples

- Assign billing1 as the password for all your system accounts.

```
passwdall
New password: <billing1>
Re-enter new password: <billing1>
```

Enter the new password (billing1), and then verify the password by typing it again.

- Remove passwords from all system accounts.

```
passwdall -d
```

pg — page through a file

Description

Displays the contents of a file or several files, one page at a time.

Formats

pg *file* Displays the contents of a file.

pg *file file* Displays the contents of multiple files, one after the other.

Movement keys

Use the keys in the chart below to move through the file.

Key(s)	Function
RETURN	move forward one screen
- (hyphen)	move back one screen
+*n*	move forward *n* screens
-*n*	move back *n* screens
+*n*d	move forward *n* lines
-*n*d	move back *n* lines
/*term*	search for *term*
?*term*	reverse search for *term*
h	display help
q	quit and return to the shell prompt

Example

- Display the office file, one screen at a time.

```
pg office
```

ping — check remote system status

Description

Determine if a system is operational on your network. If the system is active, **ping** displays "*system* is alive". If after 20 seconds the system does not respond, **ping** displays "no answer from *system*".

Formats

ping *system*
Determines whether a system is currently active on your network. The *system* is a system name from your /etc/hosts file.

ping *address*
Determines whether the system at the specified internet address is active.

Examples

- Find out if the system called elvis is working.

```
ping elvis
```

```
elvis is alive
```

- Find out if the system with internet address 192.9.205.41 is working.

```
ping 192.9.205.41
```

```
192.9.205.41 is alive
```

pr — format a file for printing

Description

Formats a file in a standard page size and displays it on the screen. You can adjust the page size, header, columns, and line spacing with **pr** options. **pr** is frequently used to format a file before sending it to be printed (**lp**).

The standard page size is 66 lines (including 10 lines of header and trailer output) by 72 columns.

Formats

pr ***file***	Paginates and displays the file.
pr -n ***file***	Paginates the file and numbers the lines on each page.

Examples

- Paginate and view the sales file.

```
pr sales.file | more
```

- Paginate and print the payroll.c program with numbers on each line.

```
pr -n payroll.c | lp
```

ps — list processes

Description

Displays a list of active processes. Active processes include those that are running, sleeping, waiting, or frozen. The **ps** command is useful when you want to terminate a process and you need to find out the process ID (PID) number.

Formats

ps	Lists active processes on the current terminal.
ps -e	Lists every active process on your system.
ps -a	Lists some important processes on your system. This list doesn't include process group leaders (shells, OPEN LOOK) and processes not associated with a terminal.
ps -u ***user***	Lists all active processes associated with the user.
ps -f	List processes in the long format.

Example

- List all the active processes on your system, using the long format.

```
ps -ef
```

```
UID     PID  PPID   C    STIME     TTY      TIME  COMD
root      0     0   0   Oct 20    ?        0:01  sched
joe       1     0   0   Oct 20    ?        1:11  /sbin/init
root      3     0   0   Oct 20    ?        0:00  pageout
```

pwconv — update the hidden password file

Description

Takes user information from the /etc/passwd file and creates a corresponding password line in the /etc/shadow file.

After you update the /etc/passwd file, typing **pwconv** "hides" password information (username, password, password aging information) in the /etc/shadow file; information on a user's password can no longer be found in the /etc/passwd file. After running **pwconv**, all passwords turn into x's in the /etc/passwd file.

Restrictions

You must be root to use this command.

Format

pwconv	Takes user information from /etc/passwd and adds a corresponding line to /etc/shadow.

Example

- Convert the /etc/passwd from a previous version of UNIX to System 5 release 4.

```
pwconv
```

pwd — print working directory

Description

Displays the full pathname of your current directory.

Format

pwd — Displays the pathname of your current working directory.

Example

- Check your current working directory.

```
pwd
```

```
/home/joe
```

rcp — remote copy

Description

Copies files or directories between different systems on a network.

Restrictions

You must have a .rhosts file in your home directory on the remote system. The .rhosts file should include the name of the local system where you are issuing the **rcp** command.

Note

If you do not specify the new filename, **rcp** uses the original filename. If you do not specify the destination directory, **rcp** uses the user's home directory.

Formats

rcp ***system:file newfile***

Copies a file from a directory on another system into your current directory with the new filename.

rcp ***file system:newfile***

Copies a file from your system to the specified directory and filename on the other system.

rcp -r ***directory system:directory***

Copies the directory on your system to the specified directory on the other system.

Example

- Copy /home/joe/birds from the crunch system to your system, and change the file name to zoo.

```
rcp crunch:/home/joe/birds zoo
```

rdb — define disk partitions

Description

Performs three disk partitioning tasks:

- creates a basic partition table
- displays information about current partitions
- adds custom formatting features to a hard disk partition

Formats

rdb ***-options device partitions***

Add formatting features to the specified partitions in the device.

rdb ***-options device***

Display information about or set formatting features for partitions on the device.

Options

Use the following options with **rdb**.

Option	Description
c	create a new rdb table
s *n*	set sectors per track to *n*
H	display information about the current partitions in a long format
p *n*	select partition *n*
b	make the selected partition non-bootable
B	make the selected partition bootable
C *n*	set the number of custom boot blocks on the selected partition to *n*
F *n*	set the file system type on the selected partition to *n*
P *n*	set the boot priority on the selected partition to *n*

Examples

- Create an **rdb** on SCSI disk 0 with 70 sectors per track and three partitions. Each partition is given a name, a starting position, and a size (in 512 byte blocks). Note that the starting position of the next partition is the sum of the starting point and size of the previous partition (140 + 178710 = 178850).

```
rdb -s 70 -c /dev/dsk/c0d0s0 Root 140
178710 Swap 178850 18900 Boot 197750 7280
```

- Display information about current partitions on SCSI disk 0, including name, starting point, length, and size.

```
rdb -H /dev/dsk/c0d0s0
```

```
    Name      Start    Length     Size

1: Unix_Root    1281    80224      38 meg
2: Unix_Swap  180352    20480      10 meg
3: Unix_Boot  200832     4096       2 meg
```

- Make partition 3, on SCSI disk 0, a bootable partition with boot priority 2.

```
rdb -p 3 -B -P2 /dev/dsk/c0d0s0
```

- Make partition 3 bootable with 2 custom boot blocks and set its file system type to DOS\0 for use with AmigaDOS.

```
rdb -p 3 -B -C 2 -F 0x444f5300
                    /dev/dsk/c0d0s0
```

- Make partition 1 non-bootable and set its file system type to UNI\0 (0x554e4900 is hex code for UNI\0) for use with Amiga UNIX.

```
rdb -p 1 -b -F 0x554e4900 /dev/dsk/c0d0s0
```

rlogin — remote login

Description

Logs in to another system from your system. If your username is in the remote system's /etc/passwd file and you have a proper .rhosts file on that system, you can log in to the remote system without entering your username and password.

Note

You disconnect from the remote system with the **exit** command or by pressing CTRL-D.

Format

rlogin ***system*** — Logs you in to another system from your system.

Examples

- Log in to the crunch system without leaving your desk. You have a user account but no .rhosts file on crunch.

```
rlogin crunch
password:
```

- Log in to the elvis system. Your home directory on elvis has a .rhosts file with the name of your current system in it.

```
rlogin elvis
UNIX System V Release 4.0 AT&T Amiga
utopia
Copyright (c) 1984, 1986, 1987, 1988 AT&T
All Rights Reserved
Last login:  Sun Oct 21 17:19:06 from narnia
System V Release 4.0          Amiga Version Release 1
$
```

rm — remove

Description

Removes files and directories.

Formats

rm ***file***	Deletes a file from your current directory.
rm ***file file***	Deletes multiple files from your current directory.
rm -ir ***directory***	Deletes a directory and all the files in it. The **-i** option asks you to confirm whether or not each file is to be deleted. Answer **y** (yes) to remove the file. If you decide to keep any of the files, you cannot delete the directory.

Examples

- Remove the pets file.

```
rm pets
```

- Delete some of the files in the current directory, but not all of them.

```
rm -i *
```

```
rm: remove  /home/joe/junk:  (y/n)? y
rm: remove /home/joe/memos:  (y/n)?n
rm: remove /home/joe/sales:  (y/n)?y
rm: remove /home/joe/sched:  (y/n)?n
rm: remove /home/joe/zoo:  (y/n)?y
```

rmdir — remove directory

Description

Removes empty directories.

Restrictions

The directory must be empty before you can use **rmdir**. If the directory has files, use the **rm -ir** command instead.

Formats

rmdir ***directory*** Deletes a directory.

Examples

- Delete the pets directory from your current directory.

```
rmdir pets
```

- Delete the bird and fish directories from your current directory.

```
rmdir bird fish
```

- Delete the /home/joe directory.

```
rmdir /home/joe
```

rn — read news

Description

Reads messages distributed by the Usenet news system.

The news system operates at three levels: newsgroup selection, article selection, and page selection. Each level has its own commands and help menu.

Formats

rn — Access the news system.

rn -c — Check for unread news messages without starting the news system.

Key(s)	Function
RETURN	read the next message
=	list articles in the news group
h	help
L	list subscribed news groups
s	save the article in a news directory
q	quit **rn**
u	unsubscribe from a news group

Example

- Start the news system.

```
rn
```

rwho — list remote users

Description

Displays a list of all users on all systems that broadcast **rwhod** status messages on your network. Systems broadcasting **rwhod** status messages in the past five minutes appear on an **rwho** list; systems that don't have the **rwhod** daemon running won't appear on the **rwho** list. If a user has not typed anything for a minute or more, **rwho** reports the user as being idle.

Restrictions

You cannot use **rwho** until you start the **rwhod** daemon by removing the comment sign (#) from the /etc/rc2.d/S72rwhod file.

rwhod uses a lot of resources, so your network response time will be slow if you choose to use this program.

Formats

rwho	Displays the status of active users on all systems that use **rwhod** on your network.
rwho -a	Reports on all users, even if they have been inactive for more than an hour.

Example

- Check the list of active users.

```
rwho -a
```

SC — spreadsheet calculator

Description

Invokes a spreadsheet program. You can use this program to create spreadsheets similar to those created using PC spreadsheet software.

Functions

Function	Steps
get help	press **?**
enter data	move to a cell, then press = followed by the numeric data
enter a formula	move to a cell, then press = followed by the coordinates of each cell you want included in the formula (separated by the appropriate arithmetic signs)
enter text	move to a cell, then press < followed by the text (to left justify the text), or press > or " followed by the text (to right justify the text)
edit data	move to a cell, press **e**, then edit the data
edit text	move to a cell, press **E**, then edit the text
delete data/text	move to a cell and press **x**
save to a file	press SHIFT-P
quit	press **q**

Format

sc

Start the spreadsheet program.

sed — edit a file or stream from a command line

Description

Invokes the stream editor, which allows you to perform edits on an entire file or pipe through a single command line.

Restrictions

The output from **sed** displays the edited file, but the actual file is not updated. You have to use the redirection (>) command to save changes to a new file.

If you include spaces in a command or between a range and a command, surround the range and command with quotes. Use quotes also for multiple line commands (**a**, **i**, **c**).

Formats

sed ***command file***

Performs the editing command on the entire file.

sed ***rangecommand file*** or

sed ***'range command' file***

Performs the editing command on a range of lines. The range format can be:

- */term/* for all lines including the term
- *linenumber* for a specific line
- *linenumber,linenumber* for a range of lines.

You can perform most of the same editing functions as **vi**; the chart on the following page lists some of the **sed** editing commands.

Command	Description
a	append the term
i	insert the term
c	change the phrase
s/*term1*/*term2*/	substitute *term1* with *term2*
d	delete

Follow the **a**, **i**, and **c** commands with a backslash and press RETURN. Except for the last line, follow each line of new text with a backslash, then put the filename on the last line.

Examples

- Substitute Monday with Tuesday in the meeting file.

```
sed s/Monday/Tuesday/ meeting
```

- Delete the fifth line of the office procedures file.

```
sed 5d office.proc
```

- Delete all lines that include the term "nocturnal" from the zoo file.

```
sed '/nocturnal/ d' zoo
```

- Change line 4 of the snacks file to "potato chips".

```
sed '4 c\
>potato chips' snacks
```

- Search file1 and replace every occurrence of "projected" with "final". Redirect the output to file2 (a new file), then rename file2 as file1 so that file1 is updated.

```
sed s/projected/final/g file1 > file2;
  mv file2 file1
```

set — display or set shell variables

Description

Displays and sets shell variables. To make changes available to other programs called from your shell, use the **export** command.

Shell variables are terms you can use in command lines to symbolize information. The start-up file (.profile) loads the values of the variables when you log in to the system. To make changes permanent, edit .profile.

When you use a shell variable in a command, type a **$** before it and type the variable in upper-case letters (for example, $HOME).

Note

set is similar to the **env** and **setenv** commands; read those *Command Reference* pages for more information.

C shell notes

The C shell allows you to change shell variables with the **set *variable=value*** command; type the variables in lowercase letters. To change environment variables, use **setenv** instead of **set**. To make the changes permanent, edit the C shell startup file (.profile).

set, cont.

Formats

set	Displays the values of all shell variables.
variable=value	Changes the value of the variable.
export ***variable***	Changes the value of the variable for other shell programs as well as the shell itself. Type this command at the end of the assignment command line or on a line of its own.

Command differences between shells

Action	ksh or sh	csh
display shell variables	**set**	**set**
change shell variables	*variable=value*	set *variable=value*
display environment variables	**env**, **printenv**	**env**, **printenv** or **setenv**
change environment variables	*variable=value*; **export** *variable*	**setenv** *variable value*

Examples

- Display the shell variables.

```
set
```

- Change the value of the EDITOR variable to **emacs** in the K shell and make the change available to all programs.

```
EDITOR=emacs; export EDITOR
```

- Change the value of the EDITOR variable to **emacs** in the C shell and make the change available to all programs.

```
setenv editor=emacs
```

setenv — display or set shell variables

Description

Displays, sets, and automatically exports environment variables. Environment variables are shell variables that are passed to programs like **mail**, **elm**, and **vi** for use in their processing.

Refer to the *Special notes on shells* section of this chapter for more information about shells.

Note

This command is similar to the **env** and **set** commands; read those *Command Reference* pages for more information.

Shell restrictions

This command works only in the C shell.

Formats

setenv Displays the values of variables.

setenv ***variable=value***

Changes the values of variables in the C shell and in all programs called by the shell.

Example

- Change the value of the EDITOR variable to **emacs** in the C shell and make the change available to all programs.

```
setenv EDITOR=emacs
```

shutdown

shutdown the system

Description

Shuts down the system; this command also allows you to delay the shutdown and change to any init state.

Restrictions

You must be root and must be in the root directory (/) to shut down the system.

Format

shutdown **-*y*** **-g *n*** **-i *initstate***

Sends a message and shuts down the system so you can turn off the power to the computer safely. The **-y** option answers "yes" to all shutdown questions. The **-g** option indicates the grace period before shutdown (in *n* seconds). The **-i** option specifies which init state to put the system in next. (The default init state is single-user mode).

Examples

- Shutdown the system after 900 seconds (15 minutes), then reboot (init state 6).

```
shutdown -y -g 900 -i 6
```

- Shutdown the system completely so you can turn the power off.

```
shutdown -y -i 0
```

sioc — screen I/O control

Description

Performs device-specific control functions. **sioc** is commonly used to change the characteristics of your screen display, including font size, window size, resolution. For more information on changing these characteristics, refer to the *Special features of Amiga UNIX* chapter.

Formats

sioc setfont
Resets the default font on your display.

sioc setfont ***fontfile***
Changes the font on your display to the one specified by *fontfile*.

sioc setdeffont ***fontfile***
Changes the system default font on your display to the one specified by *fontfile*.

sioc winsize
Displays the number of lines and columns on your display.

sioc setkmap ***kmapfile***
Changes the keyboard map for the current terminal.

sioc setdefkmap ***kmapfile***
Changes the system default for the keyboard map.

The font and keyboard map files are in the /usr/amiga/lib/font and /usr/amiga/lib/kmap directories respectively.

Examples

- Change your font for the current virtual screen session to topaz11.

```
sioc setfont /usr/amiga/lib/font/topaz11
```

- Change the default value for your keyboard map to usa2.

```
sioc setdefkmap /usr/amiga/lib/kmap/usa2
```

sleep — suspend the shell

Description

Suspends the shell for a specified number of seconds. Usually, the **sleep** command is used in conjunction with another command.

Format

sleep *x* — Pause for *x* seconds.

Examples

- Display a list of the current directory's contents in 15 seconds.

```
sleep 15; ll
```

```
drwx------  2  joe  mgt   32  Oct   4  10:07 Mail
lrwxrwxrwx  1  joe  mgt   10  Oct 20  14:55 games->/usr/games
drwxr-xr-x  2  joe  mgt   32  Oct 18  12:34 memos
-rw-r--r--  1  joe  mgt   17  Oct 10  14:30 meeting1
```

- Start a loop program that displays the message "I'm waiting" every 25 seconds and put the process in the background.

```
while sleep 25; do echo "I'm waiting";
                done &
```

```
[1]     336
                    (after 25 seconds...)
I'm waiting
                    (after another 25 seconds...)
I'm waiting
```

sort — sort lines of files

Description

Sorts the lines of a file (or files) in alphabetical order. Blanks, special characters, numbers, and uppercase letters are listed first unless you change the sort options.

Formats

sort ***file***

Sorts a file alphabetically and displays the results on the screen.

sort -o ***newfile file***

Sorts a file and puts the results in a file. The **-o** option puts the results of the sort into a file. The new file can have the same name as the file being sorted.

sort -f -o ***newfile file***

Sorts a file, ignores capitalization, and puts the results in a file. The **-f** option tells **sort** to merge upper and lower case letters.

Examples

- Sort the bird and fish files, and place the results in a file named zoo.

```
sort -o zoo bird fish
```

- Sort the lab file, ignore capitalization, and call the file by the same name.

```
sort -f -o lab lab
```

stty — set terminal options

Description

Sets or displays input/output options for your terminal.

Formats

stty	Causes some options set for your terminal to be displayed.
stty -a	Causes all terminal option settings to be displayed.
stty *keyterm key*	Changes the setting for a command key sequence.

Restrictions

Don't try to use the same key sequence for more than one operation (for example, if the BACKSPACE key is assigned as the erase key, don't try to assign it as the interrupt key also).

Examples

- Look at all your terminal options, including baud rate, command key settings, etc.

```
stty -a
```

```
speed 38400 baud;
rows = 25; columns = 80; ypixels = 200; xpixels = 640;
intr = ^c; quit = ^\; erase = ^h; kill = ^u;
eof = ^d; eol = <undef>; eol2 = <undef>;
```

- Change the interrupt command key to CTRL-V.

```
stty intr CTRL-V
```

- Change the erase key from the BACKSPACE key to the DEL key.

```
stty erase DEL
```

tail

display the end of a file

Description

Displays the last 10 lines of a file. You can also specify the number of lines to display.

Formats

tail ***file***	Displays the last ten lines of a file.
tail ***-n file***	Displays the last *n* lines of a file.
tail ***+n file***	Displays the file starting at the *n*th line.

Examples

- Display the last 10 lines of the test file.

```
tail test
```

- Display the last thirty lines of the sample file.

```
tail -30 sample
```

- Display the sample file starting at the fifteenth line.

```
tail +15 sample
```

talk — exchange screen messages

Description

Allows users to "talk" with one another on their terminal screens. Either you or the other user can terminate the **talk** program by pressing CTRL- C.

Restrictions

The user you want to "talk" to must be logged in to the system, must have **mesg** set to yes, and must reply via the **talk** program.

Formats

talk ***username*** Exchanges screen messages with a user on the same system.

talk ***username@system***

Exchanges screen messages with a user on a different system.

Examples

- Jane wants to "talk" to Joe; they are both on the same system.

```
talk joe
```

```
[Waiting for your party to respond]
```

- Joe receives Jane's request to **talk** and responds.

```
Message from Talk_Daemon@utopia at 9:56 ...
talk:  connection requested by jane@utopia.
talk: respond with:  talk jane@utopia
```

```
talk jane
```

tar

create tape archive

Description

Copies files or directories to or from the hard disk, a floppy disk, or a tape. **tar** can also list files stored in **tar** format.

Restrictions

To copy a file or directory to a floppy disk, you must use a formatted disk. The **tar** command does not format the disk. To format a floppy disk, refer to the **fdfmt** command summary.

tar doesn't copy empty directories or special files.

If you don't list a file or directory in the command, **tar** extracts or lists everything on the device.

Formats

tar ***-cvf backupdevice file(s)***

Copies (**c**) files or directories to the device. **v** is the option for verbose message and **f** is the option for file format.

tar ***-xvf backupdevice file(s)***

Extracts (**x**) files or directories from the device.

tar ***-tvf backupdevice file(s)***

Creates a table list (**t**) of the contents of the device.

tar, cont.

Examples

- List all the files and directories on a tape in an external tape drive.

```
tar -tvf /dev/rmt/4
```

```
tar: blocksize = 20
drwxrwxrwx999/1 0    Sep  5 11:54 1990  .
drwx------999/1 32   Oct  4 10:07 1990  ./Mail
-rw-r--r--999/1 17   Oct 10 14:30 1990  ./meeting1
```

- Extract the to.do file from a backup floppy disk.

```
tar -xvf /dev/dsk/fd0 to.do
```

- Copy all the files from the floppy disk onto the current directory of your hard disk.

```
tar -xvf /dev/dsk/fd0
```

- Make a backup copy of the current directory.

```
tar -cvf /dev/rmt/4 .
```

tee — copy output to two places

Description

Causes the results of a command (standard output) to be copied into a file at the same time it is being displayed on a screen. Usually, the **tee** command is part of a pipeline (see the examples below).

Format

tee ***file***	Copies standard output into a file.
tee -a ***file***	Adds standard output to the end of the file instead of creating a new file.

Examples

- Display the **man** page for the **ls** command and add the text to the end of a file named mpages.

```
man ls | tee -a mpages
```

- Print the **man** page for the **ls** command and save it in the ls.file.

```
man ls | tee ls.file | lp
```

telnet — log in to a remote system

Description

Logs in to a remote system over a network even if it uses a different operating system or another version of UNIX. **telnet** communicates with a remote system using the TELNET protocol. Once connected, use the remote system as if you were directly logged in to it.

Formats

telnet

Enters command mode and displays a command prompt (telnet>).

telnet ***remotesystem***

Connects directly to the remote system and displays a login prompt.

telnet commands

Here's a list of commands you can use while in **telnet**.

Command	Description
close	close the current connection
open	connect to another system
quit	exit **telnet**
?	display help info

Example

- Access the remote system named crunch, then log in.

```
telnet crunch
```

```
Trying 192.9.180.128
Connected to crunch.
Escape character is '^]'.

UNIX(r) System V Release 4.0 (crunch)
login:
```

tty — display terminal device name

Description

Displays the pathname of your current screen.

Format

tty — Displays the pathname of your current screen.

Example

- Check the pathname of your active screen.

```
tty
```

```
/dev/term/con3
```

type — display the pathname of a command

Description

Displays the full pathname of a command; tells you where a command resides in the directory hierarchy, if that command is in your path.

C shell note

Use the **which** command rather than **type** in the C shell.

Format

type ***command*** Displays the pathname of the specified command.

Examples

- Display the pathname of the **more** command.

```
type more
```

```
more is /usr/bin/more
```

- Find out where your **psf** command comes from.

```
type psf
```

```
psf is an alias for ps -f
```

uname — set or display system name

Description

Displays your system name and version; also can set your system name.

Restrictions

You must be root to change your system name.

Formats

uname — Displays only the system name.

uname -S *system* — Sets the name of your Amiga to the specified system name.

uname -a — Displays the system name and the operating system release, name, and version.

Examples

- Change your system name to utopia.

```
uname -S utopia
```

- Display the operating system version and release.

```
uname -a
```

```
UNIX System V utopia 4.0 Release1.1 Amiga
```

uptime

displays active time

Description

Displays the length of time the system has been active.

Format

uptime Displays the current time; the number of days, hours, and minutes the system has been active, and the number of users on the system.

Example

- Find out how many days the system has been active.

```
uptime
```

```
10:05am  up 11 days, 20 mins, 6 users
```

vi

visual editor

Description

Invokes the **vi** (visual) text editor. You can create and edit files in this editor. Here are some of the commands you use with **vi**. Refer to the *Using the vi editor* chapter for more information.

Commands

Command	Description
k	move cursor up
j	move cursor down
h	move cursor left
l	move cursor right
i	start insert mode after cursor
a	start insert mode before cursor
o	start insert mode with a blank line
ESC	quit insert mode
:w	save a file
:wq	save and quit a file
:q!	quit without saving
x	delete a character
dw	delete a word
dd	delete a line
D	delete to the end of a line
u	undo the last command
cw	change a word
cc	erase a line and insert
C	change to the end of a line
dd	cut (delete) a line
*n*dd	cut (delete) *n* lines
yy	copy (yank) a line
*n*yy	copy (yank) *n* lines
p	paste below cursor
P	paste above cursor
/term	search for *term*
. (dot)	repeat last change
n	repeat search (find next *term*)

wall — write to all users

Description

Sends a message to all users who are logged in to your system.

Restrictions

To override the message state (**mesg=n**) on other users' terminals, you must log in as root.

Format

wall — Waits for you to type the message you want to send. When you finish typing the message, press CTRL-D; **wall** then sends your message.

Example

- Send a message to system users.

```
wall
The system will be down at lunch time.
CTRL-D
```

```
Broadcast message from root (console) on utopia Wed Oct 24
10:23:38...
The system will be down at lunch time.
```

who

lists users on system

Description

Lists users logged in to the system.

Formats

who — Displays all active users on the system and indicates which virtual screens they are using and the date and time they logged in.

who -uH — Adds idle time, PID, and user comments to the **who** information. The **H** option adds headers.

who am i — Displays your username, virtual screen (tty and console), and log in date and time.

Examples

- Make sure that a particular user is logged in to the system before **talk**ing to him.

  who

```
root   console      Oct 24 10:23
joe    term/con2    Oct 19 09:05
joe    term/con7    Oct 19 12:21
```

- Find out how long Joe has been idle.

  who -uH

```
NAME   LINE     TIME              IDLE  PID  COMMENTS
root   console  Oct 24 10:23      0:12  503  system console(F1)
joe    term/con2 Oct 19 09:05     .     144  F2
joe    term/con7 Oct 19 12:21     0:12  149  F7
```

- Find out which *username* you used to log in to the current screen.

  who am i

```
joe    term/con2    Oct 19 09:05
```

whodo — who's doing what

Description

Displays the process status for all users who are logged in to the system. The information displayed is similar to the information displayed by the **ps** command.

Formats

whodo — Displays the process status for each active user.

whodo *username* — Displays the process status for the specified user.

whodo -l — Lists **whodo** information in a single-line format.

Example

- Find out what processes Joe is running.

```
whodo joe
```

```
term/con2       joe          Oct 19 09:05
      term/con7      144          0:00 ksh
      term/con7      237          0:00 csh
      term/con2      667          0:00 whodo
term/con7       joe          Oct 19 09:05
      term/con7      144          0:00 ksh
```

xhost — list systems with access to your X server

Description

Identifies systems that are allowed to connect with your X server, so you can run remote X applications from other systems. Also allows you to restrict the systems that can access your X server. Permanent access is stored in the /etc/X0.hosts file, so you don't have to issue **xhost** commands each time you start X.

Formats

xhost	Lists the systems that are allowed to display to your X server.
xhost +	Allows all users to access your X server (by disabling **xhost** security).
xhost + ***system***	Adds a specific system to the list of systems allowed to access your X server.
xhost -	Limits access only to specific systems (by enabling **xhost** security).
xhost - ***system***	Removes a system from the list of systems that are allowed to access your X server.

Examples

- Allow everyone on your network to connect X applications to your X server.

```
xhost +
```

- Remove the utopia system from the list of systems allowed to access your X server.

```
xhost - utopia
```

xset — set X user preferences

Description

Sets X Window System user preferences such as the path to fonts, mouse speed, and the time before the screen saver appears.

Formats

xset -q	Displays the current settings.
xset m *acc dis*	Sets the acceleration rate and threshold distance for the mouse. *acc* multiplies the mouse speed if the mouse quickly travels *dis* pixels.
xset r off or **on**	Turns the keyboard autorepeat off or on.
xset s off or **on**	Starts or stops the screen saver if no activity occurs.
xset s *n x*	Starts the screen saver after *n* seconds. If you include *x* in the command, **xset** changes the screen saver pattern every *x* seconds.

Examples

- Start the screen saver if the server is inactive for more than 10 minutes (600 seconds).

```
xset s 600
```

- Set the mouse to move 4 times as fast as it normally would if it passes 6 pixels in a short period of time.

```
xset m 4 6
```

xset, cont.

- Display the current X Window System user settings.

```
xset -q
```

```
Keyboard Control:
  auto repeat: on  key click percent: 10  LE mask: 0
  bell percent: 50  bell pitch: 400  bell duration: 100
Pointer Control:
  acceleration: 2=2/1  threshold: 4
```

xterm

create an xterm window

Description

Creates an X Window System terminal window. You type UNIX commands into the xterm window just as you would type them into a virtual screen or another type of terminal.

Formats

xterm -sl *lines*	Sets the number of lines scrolled off the window: 64 is the default; 256 is the maximum.
xterm -n *name*	Changes the xterm window and icon titles.
xterm -bg *color*	Sets the xterm background color.
xterm -fg *color*	Sets the xterm foreground color.
xterm -fn *font*	Specifies the font you want to use. Check the directories under /usr/X/lib/fonts for available fonts; use the first part of the font name (the part before the period).

Examples

- Start an xterm with the title "xterm number 1".

```
xterm -n "xterm number 1"
```

- Start an xterm using a large font and keeping the maximum number of scrollable lines.

```
xterm -fn 8x13 -sl 256
```

Special characters

You use special characters in commands to:

- replace characters in names
- move quickly between directories
- perform special processes

Replacement characters

Use these characters to replace characters in names.

Character	Function
*	asterisk; replaces any character(s)
?	question mark; replaces a single character
[-]	brackets with hyphen; matches a range of characters
[,]	brackets with comma(s); matches a series of characters

Use these characters to move quickly between directories.

Character	Function
.	single period; current directory
..	double periods; parent directory
/	slash alone; root directory

Special operation characters

Use these characters to perform special operations.

Character	Function
&	ampersand; background process
>	greater than; redirect output to
<	less than; redirect input from
|	pipe; use the output of one process as the input for another process
!!	double exclamation points; execute the previous C shell command
'	single quotes; treat the quoted sequence as one argument
\	backslash; take the next characters literally
;	separate commands on a command line
()	parentheses; group multiple commands into one process

Additional special commands

Additional special characters include: ", `, $, {, }, !, @, and ^. Read the shell **man** pages for more information about these special characters.

Index

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Commodore®

Commodore Business Machines, Inc.
1200 Wilson Drive • West Chester, PA 19380

313203-01
A3000UX UNIX S/W

363521-03

1/91