

*Getting Started
with Domain/OS*

002348-A00

apollo

Getting Started with Domain/OS

Order No. 002348-A00

Apollo Computer Inc.
330 Billerica Road
Chelmsford, MA 01824

Copyright © 1988 Apollo Computer Inc.
All rights reserved. Printed in U.S.A.

First Printing: May 1988

This document was produced using the Interleaf Technical Publishing Software (TPS) and the InterCAP Illustrator I Technical Illustrating System, a product of InterCAP Graphics Systems Corporation. Interleaf and TPS are trademarks of Interleaf, Inc.

Apollo and Domain are registered trademarks of Apollo Computer Inc.

UNIX is a registered trademark of AT&T in the USA and other countries.

3DGMR, Aegis, D3M, DGR, Domain/Access, Domain/Ada, Domain/Bridge, Domain/C, Domain/ComController, Domain/CommonLISP, Domain/CORE, Domain/Debug, Domain/DFL, Domain/Dialogue, Domain/DQC, Domain/IX, Domain/Laser-26, Domain/LISP, Domain/PAK, Domain/PCC, Domain/PCI, Domain/SNA, Domain X.25, DPSS, DPSS/Mail, DSEE, FPX, GMR, GPR, GSR, NLS, Network Computing Kernel, Network Computing System, Network License Server, Open Dialogue, Open Network Toolkit, Open System Toolkit, Personal Supercomputer, Personal Super Workstation, Personal Workstation, Series 3000, Series 4000, Series 10000, and VCD-8 are trademarks of Apollo Computer Inc.

Apollo Computer Inc. reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should in all cases consult Apollo Computer Inc. to determine whether any such changes have been made.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF APOLLO COMPUTER INC. HARDWARE PRODUCTS AND THE LICENSING OF APOLLO COMPUTER INC. SOFTWARE PROGRAMS CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN APOLLO COMPUTER INC. AND ITS CUSTOMERS. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY APOLLO COMPUTER INC. FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY BY APOLLO COMPUTER INC. WHATSOEVER.

IN NO EVENT SHALL APOLLO COMPUTER INC. BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATING TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF APOLLO COMPUTER INC. HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY OF SUCH DAMAGES.

THE SOFTWARE PROGRAMS DESCRIBED IN THIS DOCUMENT ARE CONFIDENTIAL INFORMATION AND PROPRIETARY PRODUCTS OF APOLLO COMPUTER INC. OR ITS LICENSORS.

Preface

Getting Started with Domain/OS introduces you to the basic concepts needed to understand and use Domain/OS. It teaches you how to use the keyboard, how to manage the information displayed on your screen, and how to edit text. There is also a discussion of the available computing environments.

You need not be an experienced programmer to use this manual. In fact, we've written the material for users with little experience using other computers. We define terms carefully and avoid jargon whenever possible.

This manual emphasizes "learning by doing." It contains numerous examples you can try while you read.

If you are an experienced programmer, this manual provides you with an easy-to-read overview of Domain/OS. After you read it, you'll be familiar with the terminology we use throughout our documentation.

We've organized this manual as follows:

PART I:	
Chapter 1	Introduces Domain/OS. Explains how to use this manual.

Chapter 2	Explains how to use the keyboard, and how to log in and log out.
Chapter 3	Describes how to manage the information on your display.
Chapter 4	Describes how the system organizes information.
Chapter 5	Explains how to create, edit, read, and print text files.
PART II:	
Chapter 6	Introduces the SysV environment.
Chapter 7	Explains how to use SysV file commands.
Chapter 8	Explains how to use the Bourne shell. Discusses wildcards, redirection of input and output, and shell scripts.
PART III:	
Chapter 9	Introduces the BSD environment.
Chapter 10	Explains how to use BSD file commands.
Chapter 11	Explains how to use the C shell. Discusses wildcards, redirection of input and output, and shell scripts.
PART IV:	
Chapter 12	Introduces the Aegis™ environment.
Chapter 13	Explains how to use Aegis file commands.
Chapter 14	Explains how to use the Aegis shell. Discusses wildcards, redirection of input and output, and shell scripts.
Glossary	Defines terms used throughout our documentation.

Related Manuals

When you complete this manual, you should continue with the user's guide for whichever of the three environments you have chosen (SysV, BSD, or Aegis). These manuals—*Using Your SysV Environment* (011022), *Using Your BSD Environment* (011020), and *Using Your Aegis Environment* (011021)—contain more advanced information about the environments and describe how to perform various tasks.

The *SysV Command Reference* (005798), *BSD Command Reference* (005800), and *Aegis Command Reference* (002547) manuals contain detailed descriptions of all standard commands in each environment. The commands are arranged alphabetically for quick and easy access.

The help file `manuals` lists current revisions of all manuals for this software release.

Problems, Questions, and Suggestions

We appreciate comments from the people who use our system. To make it easy for you to communicate with us, we provide the Apollo® Product Reporting (APR) system for comments related to hardware, software, and documentation. By using this formal channel, you make it easy for us to respond to your comments.

You can get more information about how to submit an APR by consulting the appropriate Command Reference manual for your environment (Aegis, BSD, or SysV). Refer to the `mkapr` (make apollo product report) shell command description. You can view the same description online by typing

```
$ man mkapr (in the SysV environment)
```

```
% man mkapr (in the BSD environment)
```

\$ help mkapr (in the Aegis environment)

Alternatively, you may use the Reader's Response Form at the back of this manual to submit comments about the manual.

Documentation Conventions

Unless otherwise noted in the text, this manual uses the following symbolic conventions.

literal values	Bold words or characters in formats and command descriptions represent commands or keywords that you must use literally. Pathnames are also in bold. Bold words in text indicate the first use of a new term.
<i>user-supplied values</i>	Italic words or characters in formats and command descriptions represent values that you must supply.
sample user input	In examples, information that the user enters appears in color.
output	Information that the system displays appears in this typeface.
[]	Square brackets enclose optional items in formats and command descriptions.
{ }	Braces enclose a list from which you must choose an item in formats and command descriptions.
	A vertical bar separates items in a list of choices.
< >	Angle brackets enclose the name of a key on the keyboard.

CTRL/

The notation CTRL/ followed by the name of a key indicates a control character sequence. Hold down <CTRL> while you press the key.

. . .

Horizontal ellipsis points indicate that you can repeat the preceding item one or more times.

.
. .
. . .

Vertical ellipsis points mean that irrelevant parts of a figure or examples have been omitted.

————— ☒ —————

This symbol indicates the end of a chapter.

Table of Contents

Chapter 1 Introduction

Getting to Know Your Workstation	1-1
Your Keyboard	1-3
Your Display	1-3
Domain/OS	1-4
The Display Manager (DM)	1-5
The Shell	1-5
How to Use This Manual	1-5
Domain/Delphi	1-5
Our Assumptions About Your Environment	1-6
Summary	1-6

Chapter 2 Logging In

Moving the Cursor	2-1
Using the Keyboard	2-1
Using the Mouse	2-2
Logging In	2-4

After Logging In	2-5
Shell Process Windows	2-6
DM Windows	2-7
Moving the Cursor between Windows	2-7
Entering Display Manager Commands	2-8
Typing Display Manager Commands	2-8
Using the DM Function Keys	2-8
Using Control Keys	2-9
Correcting Typing Errors	2-10
Logging Out	2-11
Summary	2-11

Chapter 3 Managing Windows

Windows	3-1
Creating a Shell Process Window	3-3
Using the Shell Key	3-3
Using the DM Command <code>cp</code>	3-4
Manipulating Windows	3-4
Changing Window Size	3-4
Using the GROW Key	3-5
Using the Mouse	3-6
Moving a Window	3-7
Popping a Window	3-7
Saving the Contents of a Transcript Pad	3-8
Closing a Window and Stopping a Process	3-9
Responding to Alarms	3-9
Summary	3-10

Chapter 4 Organizing Information

How the System Organizes Files	4-1
Pathnames	4-2
The Network Root Directory	4-3
Your Node Entry Directory	4-3

Your Log-In Directory	4-3
Your Working Directory	4-4
Parent Directories	4-4
Summary	4-5

Chapter 5 Manipulating Text

Edit and Read-Only Pads	5-1
Insert and Read-Only Modes	5-1
Opening a File for Reading	5-2
Using the Mouse to Open a Read-Only Window	5-2
Changing Between Read-Only Mode and Insert Mode	5-4
Closing a Read-Only Window	5-4
Manipulating Text in a Window	5-4
Moving to the Top or Bottom of a Pad	5-4
Scrolling Vertically	5-5
Changing the Vertical Page-Scroll Amount	5-5
Scrolling Horizontally	5-6
Opening a File for Editing	5-6
Changing the Contents of a File	5-6
Correcting Errors	5-8
Selecting Text	5-8
Copying Text	5-9
Cutting Text	5-9
Pasting Text	5-10
Moving Text	5-10
Copying and Pasting Between Files	5-11
Searching for Text	5-12
Substituting Text	5-13
Undoing Previous Commands	5-13
Saving an Edit File	5-14
Closing an Edit File	5-14
Closing and Saving a File	5-14
Discarding a File	5-14

Printing a File	5-15
With the Dialogue Option	5-15
Without the Dialogue Option	5-17
Summary	5-18

Chapter 6 Introducing SysV

What Is a Shell?	6-1
Using UNIX Commands	6-1
Getting Help	6-2
Summary	6-2

Chapter 7 Working with Files and Directories in SysV

Working with Directories	7-1
Listing the Contents of a Directory	7-1
Listing the Contents of Your Node Entry Directory ...	7-2
Listing the Contents of Your Working Directory	7-3
Listing the Contents of Your Parent Directory	7-3
Your Home Directory	7-4
Changing Directories	7-4
Identifying Your Working Directory	7-5
Creating a New Directory	7-5
Pathname Symbols	7-6
Working with Files	7-8
Copying a File	7-8
Deleting a File	7-9
Using Links	7-9
Creating a Soft Link	7-9
Creating a Hard Link	7-10
The UNIX Display-Oriented Editor (vi)	7-11
The UNIX Line Editor (ed)	7-11
Summary	7-11

Chapter 8 Using the Bourne Shell

Command Format	8-1
Using Command Arguments	8-1
Using Command Options	8-2
Entering Multiple Commands on a Line	8-2
Command Line Processing	8-3
Command Search Paths	8-3
Using Wildcards	8-5
Redirecting Input and Output	8-5
Writing Output to a File	8-6
Reading Input from a File	8-6
Using Pipes and Filters	8-7
Creating Shell Scripts	8-9
Summary	8-11

Chapter 9 Introducing BSD

What Is a Shell?	9-1
Using UNIX Commands	9-1
Getting Help	9-2
Summary	9-2

Chapter 10 Working with Files and Directories in BSD

Working with Directories	10-1
Listing the Contents of a Directory	10-1
Listing the Contents of Your Node Entry Directory ...	10-2
Listing the Contents of Your Working Directory	10-3
Listing the Contents of Your Parent Directory	10-3
Your Home Directory	10-4

Changing Directories	10-4
Identifying Your Working Directory	10-5
Creating a New Directory	10-5
Pathname Symbols	10-6
Working with Files	10-8
Copying a File	10-8
Deleting a File	10-9
Using Links	10-9
Creating a Soft Link	10-9
Creating a Hard Link	10-10
The UNIX Display-Oriented Editor (vi)	10-11
The UNIX Line Editor (ed)	10-11
Summary	10-11

Chapter 11 Using the C Shell

Command Format	11-1
Using Command Arguments	11-1
Using Command Options	11-2
Entering Multiple Commands on a Line	11-2
Command Line Processing	11-3
Command Search Paths	11-3
Using Wildcards	11-4
Redirecting Input and Output	11-5
Writing Output to a File	11-5
Reading Input from a File	11-6
Using Pipes and Filters	11-7
Creating Shell Scripts	11-8
Summary	11-10

Chapter 12 Introducing Aegis

What Is a Shell?	12-1
Using Aegis Commands	12-1
Getting Help	12-2
Summary	12-2

Chapter 13 Working with Files and Directories in Aegis

Working with Directories	13-1
Listing the Contents of a Directory	13-1
Listing the Contents of Your Node Entry Directory ...	13-2
Identifying Your Working Directory	13-2
Changing Your Working Directory	13-3
Creating a New Directory	13-3
Listing the Contents of Your Parent Directory	13-4
Displaying Your Naming Directory	13-4
Changing Your Naming Directory	13-5
Pathname Symbols	13-5
Working with Files	13-7
Copying a File	13-7
Deleting a File	13-7
Using Links	13-8
Summary	13-9

Chapter 14 Using the Aegis Shell

Command Format	14-1
Using Command Arguments	14-1
Using Command Options	14-2
Entering Multiple Commands on a Line	14-3

Entering Long Command Lines	14-3
Command Line Processing	14-4
Command Search Rules	14-4
Using Wildcards	14-6
Redirecting Input and Output	14-6
Writing Output to a File	14-7
Reading Input from a File	14-7
Creating Shell Scripts	14-9
Substituting Arguments	14-10
Summary	14-11

Appendix A: Keyboard Command Summary

Glossary

Index

Figures

1-1	Examples of Apollo Workstations	1-2
1-2	The Display	1-4
2-1	Keys that Move the Cursor	2-2
2-2	A Mouse	2-3
2-3	Default Window Positions	2-6
2-4	Function Keys that Access Two Commands ..	2-9

3-1	A Window Over a Pad	3-2
3-2	Input and Transcript Pads	3-3
3-3	Changing a Window's Size	3-6
3-4	Pushing and Popping Windows	3-8
4-1	A Sample Naming Tree	4-2
4-2	Sample Pathname	4-4
5-1	An Edit Pad in Read-Only Mode	5-3
5-2	A Sample Editing Session Using the DM	5-7
5-3	The Print Menu	5-16
7-1	A Sample Naming Tree	7-2
7-2	A New Directory in the Naming Tree	7-6
7-3	Pathnames Starting with //, /, and ../	7-8
10-1	A Sample Naming Tree	10-2
10-2	A New Directory in the Naming Tree	10-6
10-3	Pathnames Starting with //, /, and ../	10-8
13-1	A New Directory in the Naming Tree	13-3
13-2	Pathnames Starting with //, /, and ../	13-6

Tables

3-1	Predefined Window Control Keys	3-5
5-1	Keys that Move the Pad under a Window ...	5-5
7-1	Pathname Starting-Point Symbols	7-7
10-1	Pathname Starting-Point Symbols	10-7
13-1	Pathname Starting-Point Symbols	13-5
A-1	Moving the Cursor	A-2
A-2	Process Control	A-3
A-3	Window Control	A-3
A-4	Creating Edit and Read-Only Pads	A-4

A-5	Moving a Pad under a Window	A-4
A-6	Editing a Pad	A-5
A-7	Searching and Replacing Text	A-6

Part I

Introducing Domain/OS

Chapter 1

Introduction

This chapter introduces Domain/OS. You'll learn about your workstation, the way the Domain/OS software works together, and about Domain/Delphi, an optional system that allows you to access manuals online.

Getting to Know Your Workstation

The Domain® system consists of two or more of our computers, called **workstations**, linked by a local-area network. Figure 1-1 shows some of the different Apollo workstations.

Each workstation can use the data, programs, and devices of other workstations. Each contains main memory, and may have its own disk or share one with another workstation.

The workstation you're using includes a keyboard and a color or monochrome display screen. Display management software lets you create several different views, or **windows**, on the screen. Each window is a separate computing environment in which you can execute programs, edit text, or read text. The system can manage many different windows at one time, with each window running its own program. You can move the windows anywhere on your screen, change their size and shape, and overlap or shuffle them as you might papers on your desk.

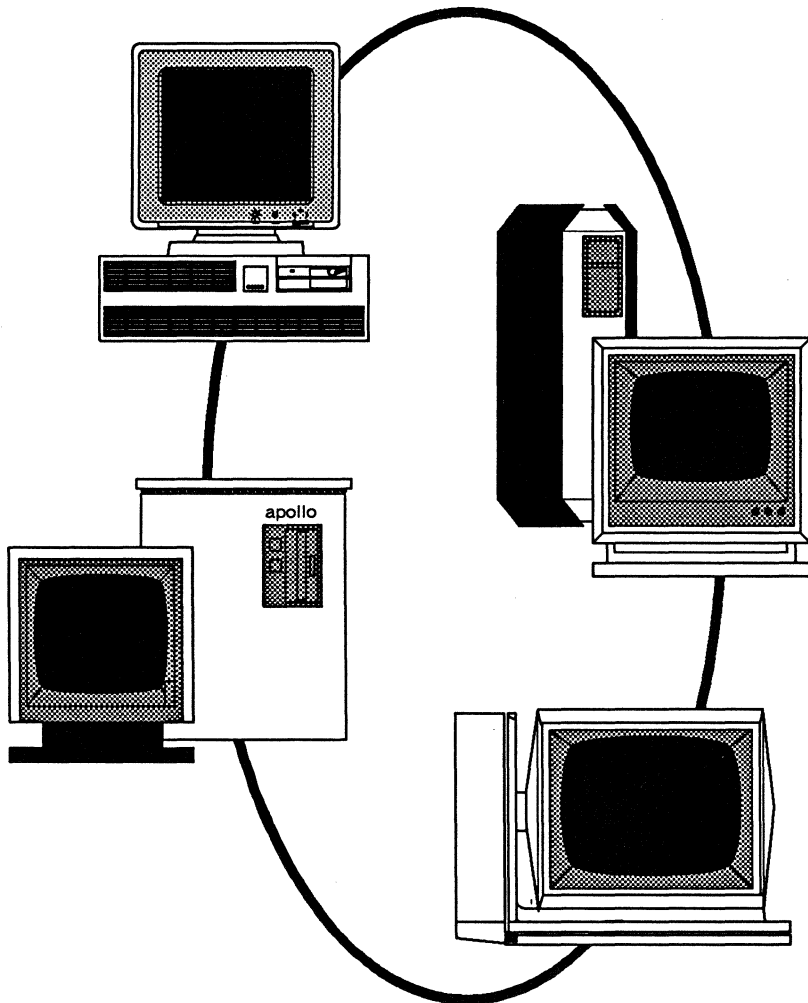


Figure 1-1. Examples of Apollo Workstations

In the back of this book, you'll find a glossary that defines terms specific to Domain/OS. The glossary also lists terms used throughout our documentation.

Your Keyboard

We provide files of standard key definitions with your system. If the keys on your keyboard don't work as we describe, your system is probably reading the wrong set of key definitions. If this happens, ask your system administrator (the person responsible for system maintenance and security at your installation) for help, or see the user's guide for your environment (*Using Your BSD Environment*, *Using Your SysV Environment*, or *Using Your Aegis Environment*).

Your Display

The display you're using resembles the **landscape** (horizontal) display in Figure 1-2. If your screen looks blank, press any key to turn on the video display. (The system automatically shuts off the video display if it is idle for more than 15 minutes.)

If your screen still looks blank after you press a key, your workstation is not running. For help with starting your workstation, see your system administrator, or refer to the operating manual for your particular model.

Notice the small blinking box in the lower left corner of your display. This is the **cursor**. The box cursor indicates where the system will display the information you type at your keyboard.

As you use the system, you will see one other cursor shape: an arrow cursor (↑). The arrow cursor appears when you use the touchpad or mouse to move the cursor. (See the section "Using the Mouse" in Chapter 2.)

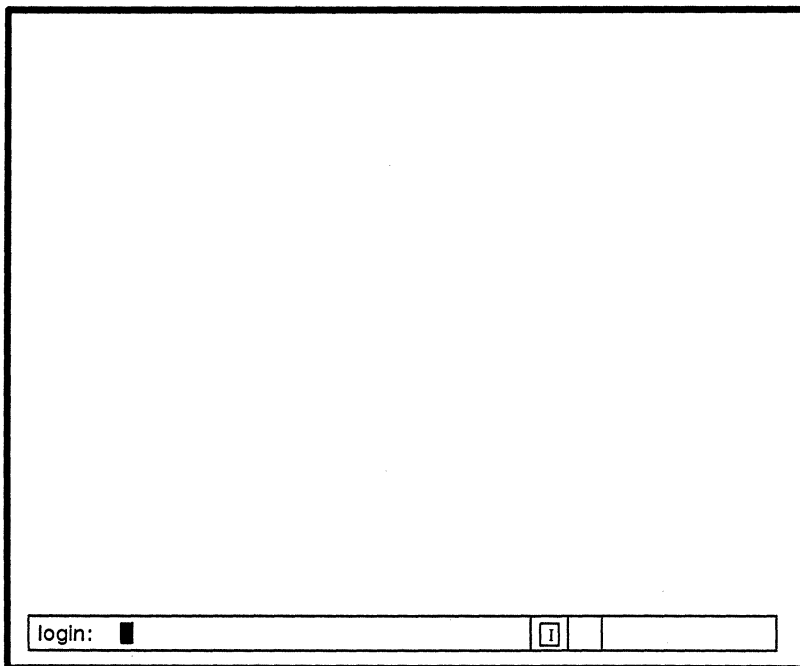


Figure 1-2. The Display

Domain/OS

Domain/OS has three environments: SysV, BSD, and Aegis. You may use one or more of these, depending on your installation.

No matter which environment you choose, you'll use a **shell** and the **Display Manager (DM)**. If you use SysV or BSD, you'll have one of the following shells: the C shell, the Bourne shell, or the Korn shell. If you use Aegis, you'll have the Aegis shell.

The Display Manager (DM)

The DM is a program that opens, closes, and moves windows on your screen. You can also use it to change other aspects of the display, such as background color and character font, and for creating and editing files. The DM also supervises the creation of computing environments (**processes**) in which you execute programs. When you log in, the DM creates a process in which a shell program is running.

The Shell

The shell is a program that provides access to traditional computing operations, such as printing documents, compiling and running programs, and monitoring system activities. The shell “listens” for commands typed at your keyboard. Shell commands invoke utilities, which are programs that perform tasks you request.

How to Use This Manual

This manual is written in four parts. The first part is for all Domain/OS users. Parts II, III, and IV are for users of specific environments. If you are a SysV user, you will use Parts I and II of this manual. If you are a BSD user, you will use Parts I and III. If you are an Aegis user, you will use Parts I and IV of this manual.

Domain/Delphi

Domain/Delphi is optional software that lets you access document pages on the screen. These document pages are identical to those found in the hardcopy manuals on your bookshelf. If you have Domain/Delphi, you may want to use it to access the information in this book online.

Our Assumptions About Your Environment

Domain/OS is very flexible and can be tailored to your particular needs. In this manual, we assume that your system is set up with the DM as the default window manager, and that you are using the default key definitions for your environment.

We assume that if you use SysV, you use the Bourne shell. If you use BSD, we assume you use the C shell. And if you use Aegis, we assume you use the Aegis shell.

Summary

This chapter introduced Domain/OS. The next chapter explains how to start using your system.



Chapter 2

Logging In

Each time you log in, Domain/OS executes programs that define your workstation's operating environment.

Moving the Cursor

Moving the cursor is the first step in learning to use the system. As you work through the examples in this book, you'll see that you must position the cursor at a specific location on your display before giving an instruction to the system.

To move the cursor, use the arrow keys or the mouse. The mouse is an optional piece of equipment; arrow keys are available on all keyboards.

As you read in the next three sections about the ways to move the cursor, try using the keys and the mouse if you have one.

Using the Keyboard

Use any of the keys highlighted in Figure 2-1 to move the cursor. To move the cursor using the arrow keys (↑ ↓ ← →), press the arrow key that points in the direction you wish to move and hold it down until the cursor reaches the desired destination.

To move to the beginning or end of the line of text beneath the cursor, use the keys labeled |← and →|.

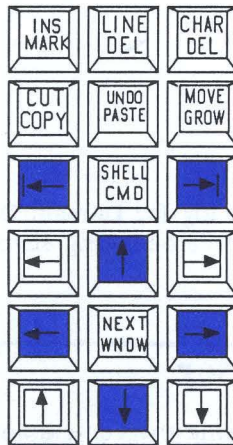


Figure 2-1. Keys that Move the Cursor

Using the Mouse

The mouse is a small device that you move across a flat surface such as your desk. (See Figure 2-2.) A ball in the base of the mouse detects motion; the mouse transmits data about the motion to the system; and the system moves the cursor.

When you move the mouse, the cursor appears as an arrow; when you stop moving the mouse, the cursor reverts to a blinking box.

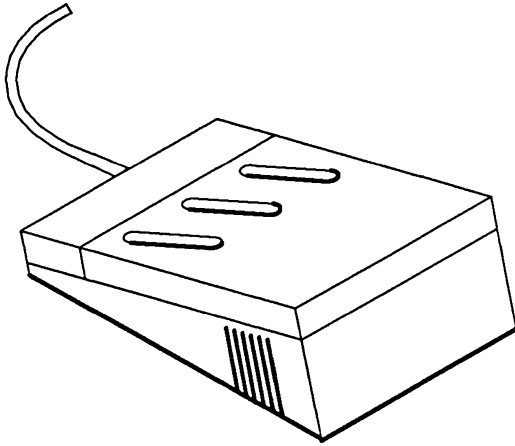


Figure 2-2. A Mouse

The mouse has three buttons that you can use to manipulate windows and read files. Chapter 3 explains how to expand, shrink, and shuffle windows with the mouse. Chapter 4 describes how to use the mouse to examine the contents of files.

NOTE: The mouse buttons are predefined to perform particular functions. However, you can redefine the mouse buttons (and the keyboard keys) to do other useful tasks. For details, see “Using the Display Manager” in the user’s guide for your environment.

Logging In

When your workstation is running and the video display is on, a log-in prompt appears at the bottom of the screen. The prompt reads

login:

When the “login:” prompt is displayed, the system is waiting for you to enter your **user ID** and **password**. If you don’t know what to enter, ask your system administrator, who defines a user account for every person authorized to use the system.

Your user account contains the name that the computer uses to identify you (user ID), as well as your password. The system uses this information to determine who can use the system and what resources they can use.

To log in, follow these steps:

1. Move the cursor to the right of the “login:” prompt, using the mouse or cursor keys, or by pressing <CMD>.
2. Enter your username as follows:

login: *username*

If your system administrator says that project and organization names are required, type these also. Use the form

login: *username.project.organization*

If you make a mistake, press <BACKSPACE>. The BACKSPACE key works like the same key on a typewriter, except that it deletes characters as it moves the cursor back toward the beginning of the line.

When you have typed the required names correctly, press <RETURN>. (Press <RETURN> whenever you wish to submit to the system the line you’ve typed.) The system accepts the line and requests your password.

3. Type your password. The system does not display the password you enter, but displays a dot for each character in the password:

Password:

If the system cannot find a user account that matches the names you supply, it repeats the log-in prompt after displaying the message

Login incorrect

If you receive this message, you are not using a valid username and password. Repeat the steps above, making certain to type the correct information. If you are still unsuccessful, ask your system administrator for help.

After Logging In

Immediately after you log in, the DM creates some windows on your screen. Figure 2-3 shows the default window positions.

In addition to presenting these windows, the DM creates processes in which one or more specified shells are running. A *shell* is a command processor that lets you run programs and utilities. The shells created depend on how your DM environment is set. For example, in Figure 2-3, the DM started a C shell.

Notice the new position of the cursor. When you logged in, you saw that the cursor position indicates where the system displays the commands you type. The cursor position also indicates which program (shell or DM) receives your commands.

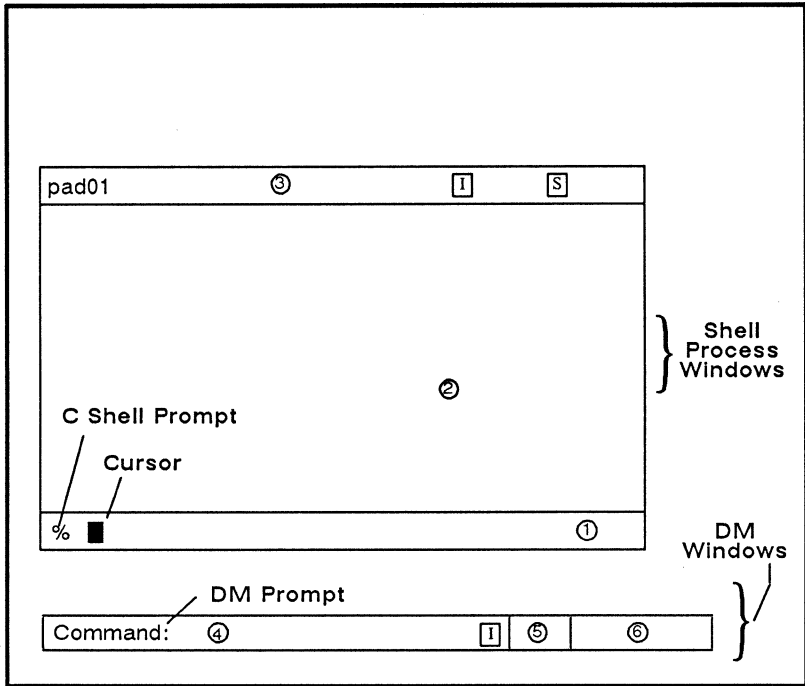


Figure 2-3. Default Window Positions

Shell Process Windows

A **process** is a running program. A **shell process window** is a window that runs a shell, allowing you to execute shell commands. A shell process window contains three parts, as shown in Figure 2-3:

1. The **process input window** for a shell contains its prompt. When you type commands in this window, the shell reads the commands and invokes the appropriate program.
2. The **process output window** (the large window above the process input window) provides a transcript, or record, of your interaction with the system. It displays your commands (after you press <RETURN>) as well as the shell's response to the commands. The response can be information you requested, a process status report, or an error message.

3. The **window legend** is the top border of the output window. It contains the window name (*padnn*) and the letters I (Insert) and S (Scroll). The letter “I” indicates that you can insert text in the input window rather than overstrike the existing command line. The letter “S” specifies that the output window scrolls automatically when new information is written to it.

The chapter “Controlling the Display” in the user’s guide for your environment explains these window mode indicators in greater detail.

DM Windows

You’ll use DM windows to enter DM commands. The DM uses three windows (as shown in Figure 2-3):

4. The **DM input window** contains the “Command:” prompt. To move the cursor next to this prompt, press <CMD>. Enter DM commands in this window. Note the mode indicator “I” to the right of the DM input window, which shows that the DM input window is in insert mode.
5. The **DM alarm window** displays a small pair of bells when a process displays a message in an output window that is hidden by an overlapping window. “Responding to Alarms” in Chapter 3 explains more about this window.
6. The **DM output window**, like the shell output window, displays messages. However, the DM does not display the commands you enter.

Moving the Cursor between Windows

To move from window to window, press <NEXT WNDW>. When you press <NEXT WNDW>, the DM invokes a command to move the cursor to the next window. Depending on the number of windows on your display, you may have to press <NEXT WNDW> more than once to move the cursor to the window you want.

Entering Display Manager Commands

There are three ways to enter DM commands:

1. Type the command in the DM input window (next to the “Command:” prompt).
2. Press a specially defined (and labeled) key, called a **DM function key**.
3. Press a **control key sequence** (<CTRL> combined with another key).

The effects of the function keys and control key sequences differ depending on which environment you use. Refer to “Using the Display Manager” in the user’s guide for your environment for an explanation of the key definitions.

Typing Display Manager Commands

Try typing a command in the DM input window. First, press <CMD> to move the cursor next to the “Command:” prompt. Now type

```
Command:  rs
```

The **rs** (refresh screen) command refreshes the screen. The display blinks as the DM clears the screen and redraws all windows.

Using the DM Function Keys

In using the DM, you usually do not type DM command names; instead, you’ll use certain single keys (DM function keys) to invoke DM commands.

Each function key highlighted in Figure 2-4 accesses two commands. For example, the function key labeled CUT and COPY lets you delete (cut) or copy text. To enter the CUT command, hold down <SHIFT> while pressing the function key. To enter the COPY command, simply press the function key.

Correcting Typing Errors

Don't worry if you make a typing mistake when you enter a command. Usually, you'll simply cause the system to display an error message on the screen. If you notice the mistake before pressing <RETURN>, correct it by using one of the keys listed here:

<BACKSPACE>

<CHAR DEL>

<LINE DEL>

<INS>

Remember that <BACKSPACE> deletes characters as it moves the cursor back toward the beginning of the line. For example, if you type `date` and then press <BACKSPACE>, the letter `e` disappears.

<CHAR DEL> deletes the character at the current cursor position. For example, if you position the cursor on the `d` in `date` and press <CHAR DEL>, the `d` disappears.

<LINE DEL> deletes the entire line, no matter where the cursor is positioned on the line.

Notice the letter "I" in the shell's window legend. The letter also appears to the right of the DM input window. It indicates that the DM and shell input windows are operating in insert mode. As you recall, insert mode lets you change command lines in the input window by repositioning the cursor and inserting characters. The rest of the line moves to the right as you insert additional characters.

To overstrike the command line, turn off insert mode by pressing <INS>. If the cursor is in the shell input window, the letter "I" in the window legend disappears. If the cursor is in the DM input window, the letter "I" following the DM input window disappears. The system then replaces existing characters with the new characters that you type. Turn insert mode back on by pressing <INS> again.

Logging Out

When you're ready to end the session, log out from the system. Logging out prevents others from accessing your user account. Log out if your workstation is in a public place.

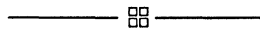
To log out, press <CMD> and type the `lo` (logout) command at the DM prompt as follows:

Command: `lo`

All your processes will stop, and all windows will close automatically. After logging you out, the DM redisplay the "login:" prompt.

Summary

After reading this chapter, you should be able to log in, interpret information on your display, move the cursor, use a shell, enter DM commands and log out. The next chapter explains how to manipulate windows.



Chapter 3

Managing Windows

This chapter explains how to use the DM to control the windows on your display. You'll learn how to view hidden information in a window, shuffle windows and change their size, open and close windows, and save the contents of a window.

Windows

Windows are areas where you view information on your display. What you view in a window can be information you type into the system, or information stored in the system. You can open and close a window, move it around on the display, change its size, and scroll through the window. You can have more than one window on your display at a time.

A window provides a view of a **pad**, or temporary file. The window can show the entire pad or only part of it, and you can have more than one window viewing the same file. Figure 3-1 illustrates a window over a pad, where only part of the pad is visible.

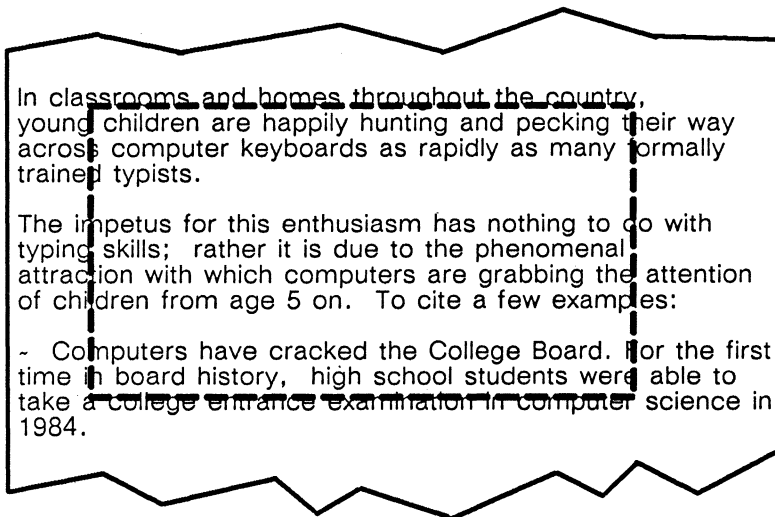


Figure 3-1. A Window Over a Pad

The dotted line in Figure 3-1 represents the window's edges. On the display, you can see only the characters inside the edges of the window. The rest of the pad is hidden. You can view the hidden parts of a pad by scrolling the pad or enlarging the window. Whether you can see an entire pad in a window depends on the sizes of the pad and window. Pads containing large amounts of text cannot be seen all at once in even the largest windows.

There are three kinds of pads: input, transcript, and edit pads. **Input pads** accept the commands you type at the keyboard. In Chapter 2, you typed commands into the DM input pad.

Transcript pads (process output pads) keep a running record of your interaction with programs; they contain both your input and the programs' output.

Edit pads allow you to view and edit text files. Chapter 5 provides information on creating and using edit pads.

Figure 3-2 shows windows onto a C shell input pad, a C shell transcript pad, and a DM input pad.

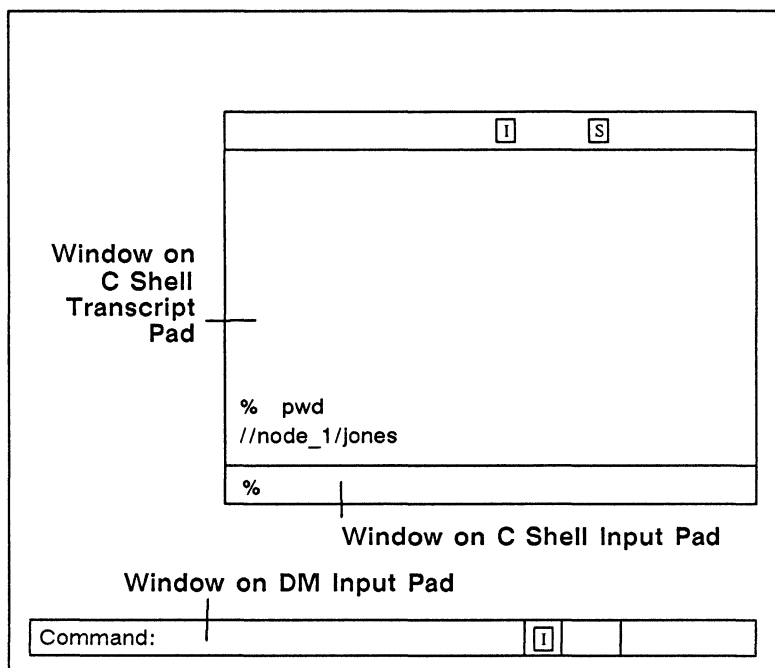


Figure 3-2. Input and Transcript Pads

Creating a Shell Process Window

Windows allow you to perform many different kinds of tasks concurrently. For example, if a shell process window is busy compiling a program, you can create a new shell process to perform another task, such as formatting a text file.

You can create a shell process window in either of two ways: using the SHELL key, or using the DM command `cp` (create process).

Using the Shell Key

Once you are logged in, press `<SHELL>` to create a shell process window. The DM places the cursor next to the shell prompt in the new process input window, and the process is ready to receive shell commands.

Using the DM Command `cp`

You can use the DM command `cp` to create a shell process window running a C shell, Bourne shell, Korn shell, or Aegis shell.

Do the following to create a shell process window:

1. Move the cursor to the DM prompt. You can move the cursor to the DM prompt using the mouse, or by pressing the CMD key.
2. Type one of the following:

<code>cp /bin/csh</code>	for a C shell
<code>cp /bin/sh</code>	for a Bourne shell
<code>cp /com/sh</code>	for an Aegis shell
<code>cp /bin/ksh</code>	for a Korn shell

The DM creates a shell process window running the shell you specified.

Manipulating Windows

The predefined window control keys listed in Table 3-1 change the size, position, and characteristics of windows on the screen. In addition, all window-control keys cause the DM to display the selected window completely if any part of it is hidden. Note that we show the equivalent mouse key where appropriate.

Changing Window Size

You can change the size of a window using the GROW key or the mouse. In either case, the DM lets you select a window corner, which you move across the screen to change the window's size.

Table 3-1. Predefined Window Control Keys

Task	Keyboard	Mouse
Shuffle windows	<POP>	Center key
Enlarge or reduce a window	<GROW>	Left key
Move a window	<MOVE>	—

Using the GROW Key

Do the following to change the size of one of the windows on your screen using <GROW>:

1. Decide which corner you want to change, and move the cursor to it. You can use the cursor keys or the mouse to move the cursor.
2. Press the GROW key. A flexible “rubberband” border appears. Figure 3-3 illustrates this flexible border.
3. Move the cursor to stretch or shrink the flexible border until it indicates the new window size you want.
4. Press <MARK>. The old window shrinks or expands according to the position of the flexible border.

If, after the flexible border appears, you decide not to change the window size, press CTRL/X.

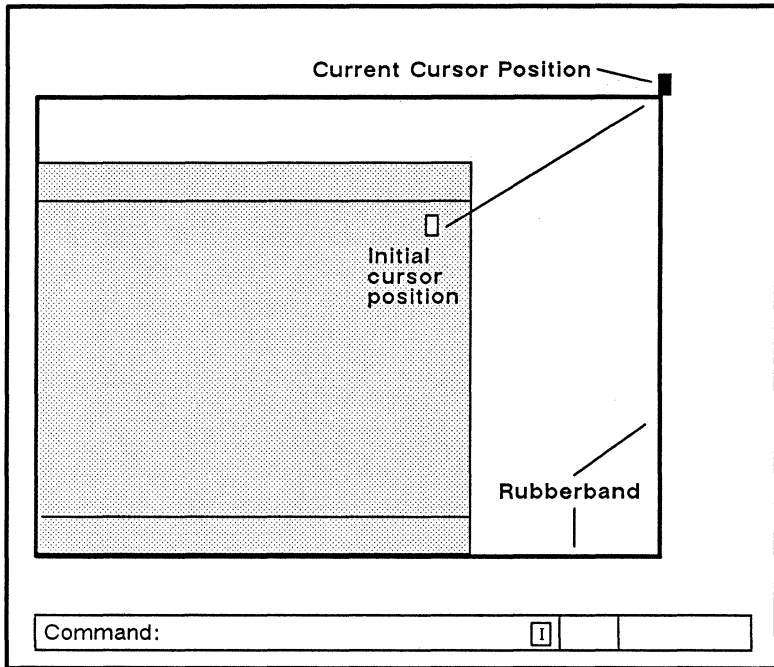


Figure 3-3. Changing a Window's Size

Using the Mouse

If you have a mouse, try using it to change window size. Follow these steps:

1. Place the cursor near the edge or corner you wish to move.
2. Press the leftmost mouse key and hold it down. The flexible border appears.
3. While you hold the left key down, move the mouse to indicate the new window size.
4. Now release the key. The DM resizes the window.

Moving a Window

To change the position of a window without changing its size, use <MOVE>. Follow these steps:

1. Place the cursor anywhere in the window.
2. Press <MOVE>. A movable copy of the window border appears.
3. Move the cursor to position this border where you want the window to appear.
4. Press <MARK>. The window then moves to its new location.

If, after the movable border appears, you decide not to move the window, press CTRL/X.

Popping a Window

Popping windows lets you display windows that are partially or completely hidden by other windows on your screen. Pressing <POP> or the center mouse button (with the cursor visible in the desired window) pops a partially hidden window to the top of the pile of windows on the screen. If the window is already in front of the other windows, <POP> sends it to the back. Figure 3-4 illustrates how <POP> works.

Try using <POP> (or the center mouse button) to shuffle the windows on your screen. (Use <SHELL> to create several windows first.) Move the cursor into a window that is partially hidden by another window. Press <POP> or the center mouse button. The window pops to the top of the pile.

Now move the cursor into a window that is completely visible. Press <POP> (or the center mouse button) again. This time the DM pushes the window to the back.

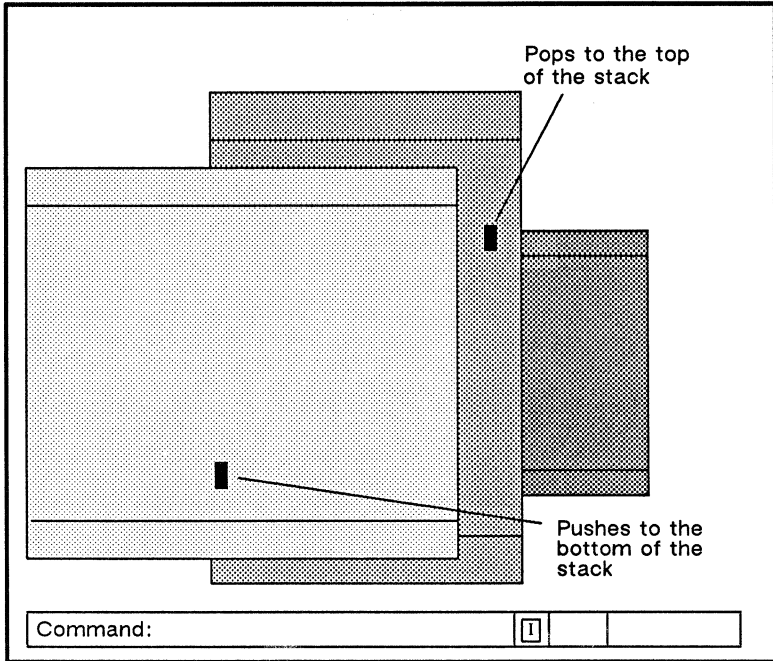


Figure 3-4. Pushing and Popping Windows

Saving the Contents of a Transcript Pad

When you've finished experimenting, you may want to save the contents of the transcript pad. The DM command **pn** (pad name) lets you save a copy of a transcript pad in a named file. Move the cursor onto the transcript pad, then press <CMD> and type the following:

Command: **pn save_file**

The **pn** command makes the temporary transcript pad a permanent file named **save_file**. The DM continues to add transcript pad output to **save_file** until you stop the process and close the windows. Refer to the *Domain Display Manager Command Reference* for more information about the **pn** command.

Closing a Window and Stopping a Process

At any time you can stop the shell processes you created and close the associated pads and windows. To do this, follow these steps:

1. Move the cursor into the process input window and type either CTRL/D (if you are a UNIX* user) or CTRL/Z (if you are an Aegis user). This control key sequence sends an end-of-file character, which stops the process, deletes the input window, and closes all pads associated with the process. The system displays

```
***EOF***  
***Pad Closed***
```

The process input window disappears from your screen, but the window to the transcript pad remains.

2. To close the window to the transcript pad, keep the cursor within the window and press <EXIT>.

Responding to Alarms

When new information appears in a partially obscured window, the DM alerts you by displaying two small bells in its alarm window. If your system has a speaker, the DM also emits an alarm tone. To respond to an alarm, use the DM command **ap** (alarm pop).

To see how **ap** works, experiment with <SHELL> and <POP> until you're comfortable manipulating windows. Now position the windows so that one window overlaps the other's input window. Use <POP> to bring the hidden window to the top. Then use the **man** or **help** command to request online information about the **cal** (print calendar) or **calendar** (print calendar) command. Type the following if you are using SysV or BSD:

```
% man cal
```

*UNIX is a registered trademark of AT&T in the USA and other countries.

Type the following if you are using Aegis:

\$ help calendar

Press <RETURN> and then quickly press <POP> again to send the top window to the bottom of the stack.

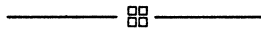
If this exercise works, the DM will produce an alarm. Enter the command **ap** in the DM input window to look at the window requesting your attention. For example, press <CMD> and type

Command: **ap**

After you press <RETURN>, the DM pops the window that needs attention. See the chapter “Controlling the Display” in the user’s guide for your environment for more information about alarms.

Summary

In this chapter, you learned how the DM controls windows, pads, and processes. The next chapter explains how Domain/OS organizes information.



Chapter 4

Organizing Information

This chapter describes how Domain/OS organizes information. You'll learn about naming trees, directories, files, links, and pathnames.

How the System Organizes Files

Domain/OS allows you to organize related information in files. We supply certain files with system software. You can create other files and determine their contents. A file might contain a memo, a program, or an illustration—anything you like.

Directories are used to organize files. Files and directories are arranged in a hierarchical structure called the **naming tree**. This chapter introduces the basic concepts you'll need to understand and use the naming tree. Figure 4-1 shows a sample naming tree.

The naming tree includes every file and directory in the network (not just those on your node). Each directory in the naming tree appears above the files and subdirectories that it contains. Naming tree components are called **objects**.

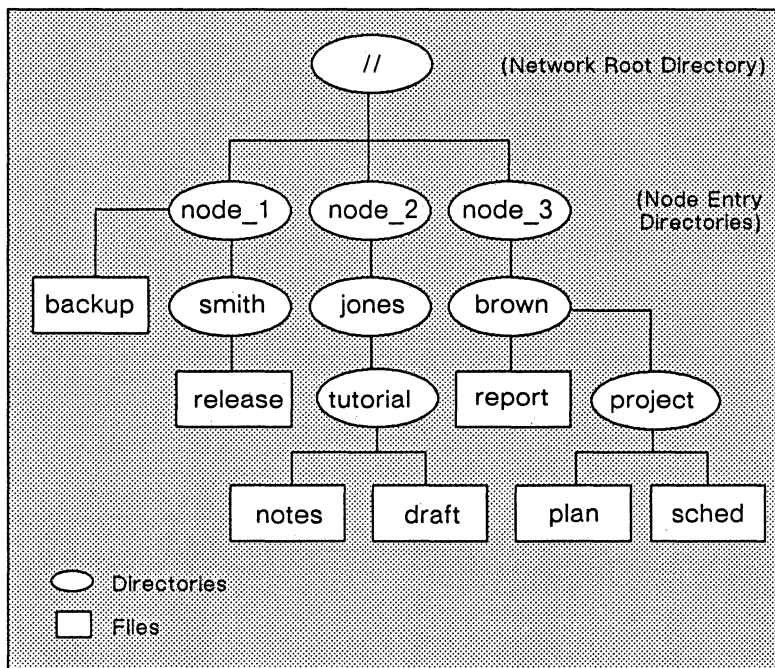


Figure 4-1. A Sample Naming Tree

In addition to files and directories, the naming tree includes a third type of object: links. A link points directly to, or contains the name of, another network object. We explain how to create and use links later in this manual. For now, think of links as a special object type that lets you take a detour from one part of the naming tree to another.

The object names used in the examples in this chapter are based on the sample naming tree in Figure 4-1. The object names in your naming tree will be different.

Pathnames

To use an object in the naming tree, you must be able to locate it. A **pathname** describes the path that the operating system must take to get from some starting point in the naming tree to a destination object.

For example, consider the file `notes` in Figure 4-1. One pathname we can use to locate `notes` looks like this:

```
//node_2/jones/tutorial/notes
```

A pathname begins with the starting point's name, includes every directory name between the starting point and the destination object, and ends with the destination object's name. A slash separates names within a pathname. An individual name (between slashes) may not exceed 255 characters. The entire pathname may not exceed 1023 characters, including the slashes.

In the above pathname, the starting point is the directory at the top of the naming tree, the **network root directory**. Double slashes (`//`) refer to the network root directory. The destination object is the file `notes`. The directories between the starting point (`//`) and the destination object are `node_2`, `jones`, and `tutorial`. Figure 4-2 highlights the path that the system follows in its search for `notes`.

The Network Root Directory

The network's top-level directory, the **root directory** (`//`), is a list of directory names. It contains the name of the top (or entry) directory for each node on the network. Thus, the root directory shown in Figure 4-2 contains three objects: the directories `node_1`, `node_2`, and `node_3`.

Your Node Entry Directory

The **node entry directory** is the top directory on each node and is a subdirectory of the network root directory.

Your Log-In Directory

The directory where the system puts you at login is called your **log-in directory**. Your user account contains your log-in directory name.

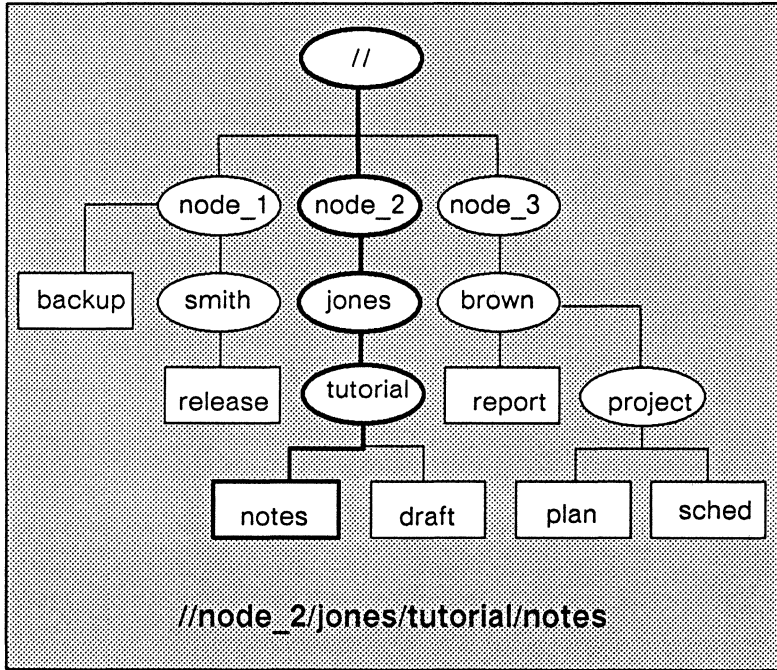


Figure 4-2. Sample Pathname

Your Working Directory

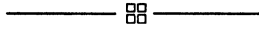
Your **working directory** is the directory in which you are currently located. This directory is sometimes referred to as your **current directory**.

Parent Directories

The **parent directory** is the directory above the working directory in the naming tree. For example, if your working directory is `//node_3/brown/project`, the parent directory is `//node_3/brown`.

Summary

This chapter described how Domain/OS organizes objects in the naming tree. The next chapter explains how to manipulate text.



Chapter 5

Manipulating Text

This chapter describes how to create, edit, and read files using DM function keys, control-key sequences, and commands. In sample editing sessions, you'll create a new file, enter text, and modify the text.

Edit and Read-Only Pads

Edit pads contain copies of files that the DM displays when you press <EDIT>. You can change text displayed on pads created with <EDIT>, and save the changes in the permanent file. **Read-only pads** contain copies of files that the DM displays when you press <READ>. You can copy text from a read-only file, but you cannot add or delete text.

Insert and Read-Only Modes

A pad is in insert mode when the window legend contains the letter "I". A window legend containing the letter "R" indicates that a pad is read-only. You can toggle between insert and read-only modes using SHIFT/<AGAIN>, as explained in detail later in this chapter.

Opening a File for Reading

To read a file, press <READ>. The cursor moves into the DM input window and the DM displays the prompt

Read file:

You must enter the name of a file that already exists. If you enter a filename that does not already exist, the DM displays the following error message:

```
(cv) filename - name not found from (stream
manager / IOS)
```

The DM displays “(cv)” because <READ> invokes the DM command `cv` (create view), which opens a window for viewing the specified text file. For more information about `cv`, see “Controlling the Display” in the user’s guide for your environment.

We’ve supplied a text file, called `sample_edit`, to help you learn about the DM’s read function. Type the file’s pathname as follows:

Read file: `/domain_examples/getting_started/sample_edit`

The DM displays the file `sample_edit`, as in Figure 5-1. Notice the letter R (for Read) displayed in the window legend. Because `sample_edit` is now displayed in a read-only window, you may not modify its contents. If you attempt to do so, the DM displays this message:

```
Text is read-only
```

The mistakes in Figure 5-1 are intentional. You’ll correct these errors later in this chapter.

Using the Mouse to Open a Read-Only Window

The right mouse key also opens files for reading. To use the mouse to open a read-only window, you need to display on your screen the name of the file you wish to read.

Typing the `ls` (list directory) command at a UNIX shell prompt, or the `ld` (list directory) command at an Aegis shell prompt, lists the names of the files in a given directory.

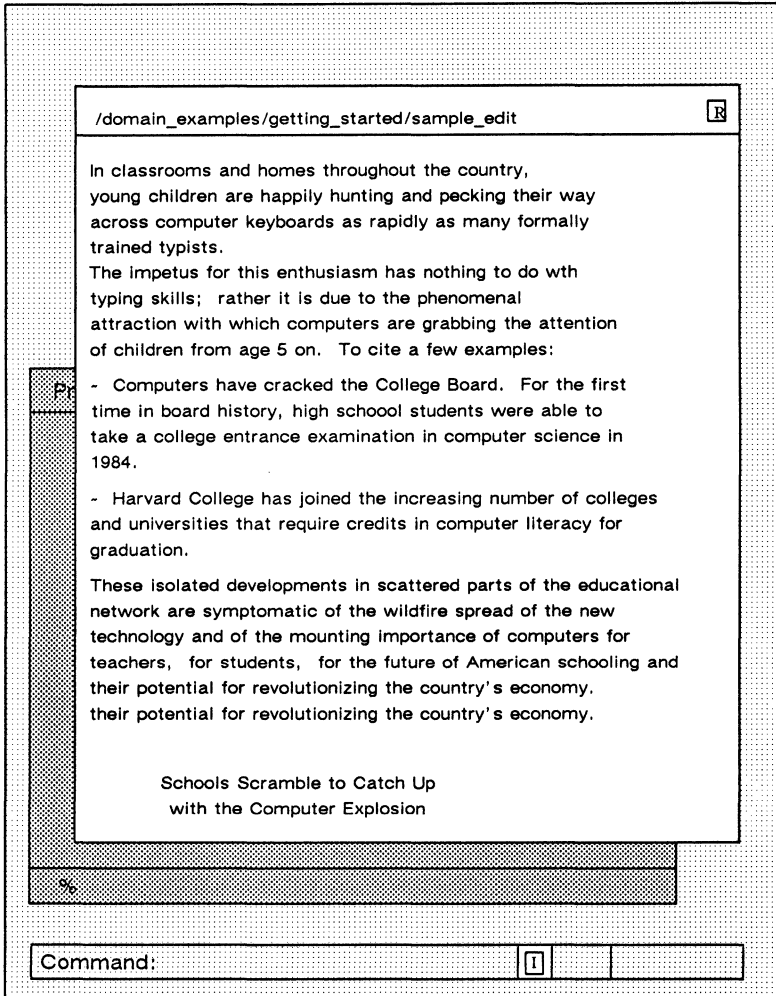


Figure 5-1. An Edit Pad in Read-Only Mode

When the name of the file you want to read appears on your screen, point to it with the cursor. When you press the right mouse key, the DM opens a read-only window for the file you selected.

Changing Between Read-Only Mode and Insert Mode

You can change a read window into an edit window and vice versa by pressing `SHIFT/⟨AGAIN⟩`. (Aegis users may alternatively use `CTRL/M`.) The window legend displays the letter “I” or “R” as appropriate.

Note that you must have permission to modify the file you specify in order to edit it. For a discussion of permissions or access rights, see the chapter “Controlling Access to Files and Directories” in the user’s guide for your environment.

Closing a Read-Only Window

To close a read-only window,

1. Move the cursor into the window.
2. Press `⟨EXIT⟩`.

The DM closes the read-only window.

Manipulating Text in a Window

The keys listed in Table 5-1 move a pad around under a window.

As you read about the keys discussed in the next three sections, try using them to move the pad containing the text of the file `/domain_examples/getting_started/sample_edit`.

Moving to the Top or Bottom of a Pad

To move the cursor to the top of the pad, press `CTRL/T`. The DM positions the cursor on the first character in the pad, scrolling the pad if necessary. To move the cursor to the bottom of the pad, press `CTRL/B`.

Table 5-1. Keys that Move the Pad under a Window

Task	Predefined Key
Move to first character in a pad	CTRL/T
Move to last character in a pad	CTRL/B
Move pad by pages	<div style="border: 1px solid black; width: 20px; height: 20px; margin: 0 auto; display: flex; align-items: center; justify-content: center;">↑</div> <div style="border: 1px solid black; width: 20px; height: 20px; margin: 0 auto; display: flex; align-items: center; justify-content: center;">↓</div>
Move pad by lines	SHIFT/↑ SHIFT/↓
Move pad by characters	SHIFT/→ SHIFT/←
Move pad by columns	<div style="border: 1px solid black; width: 20px; height: 20px; margin: 0 auto; display: flex; align-items: center; justify-content: center;">←</div> <div style="border: 1px solid black; width: 20px; height: 20px; margin: 0 auto; display: flex; align-items: center; justify-content: center;">→</div>

Scrolling Vertically

The SHIFT/↑ and SHIFT/↓ keys scroll a pad vertically up and down one line, respectively. The cursor position does not change relative to the pad; the pad simply slides by under the window.

The keys

↑

 and

↓

 scroll a pad vertically in units of half the height of the window.

Changing the Vertical Page-Scroll Amount



Use the DM command **pp** (pad page) to modify the amount of text that the boxed arrow keys scroll. To set the number of pages to scroll, type the command followed by the number of pages that you want to scroll. (A page is defined as the smaller of either the num-

ber of lines that fit in the window, or the number of lines between the bottom of the window and the next form feed.)

For example, to scroll three pages, type

Command: **pp 3**

Scrolling Horizontally

The  and  keys scroll a pad left and right under a window in 10-character increments.

The left and right scrolling keys (SHIFT/← and SHIFT/→) scroll a pad in single-character increments.

Opening a File for Editing

To open a file for editing, press <EDIT>. The cursor moves to the DM input window and the DM prompts you to specify the name of the file:

Edit file:

If you enter the name of an existing file, the DM opens that file for editing. If you enter a filename not in current use, the DM creates a new file.

Changing the Contents of a File

We'll use `sample_edit` again to learn about the DM's editing functions. To open `sample_edit`, type the following pathname next to the DM edit prompt:

Edit file: `/domain_examples/getting_started/sample_edit`

When you press <RETURN>, the DM opens an edit pad containing the text of the file `sample_edit` as shown in Figure 5-2. Notice that the pathname appears in the window legend. We've included a few mistakes that we'll use to demonstrate some editing functions.

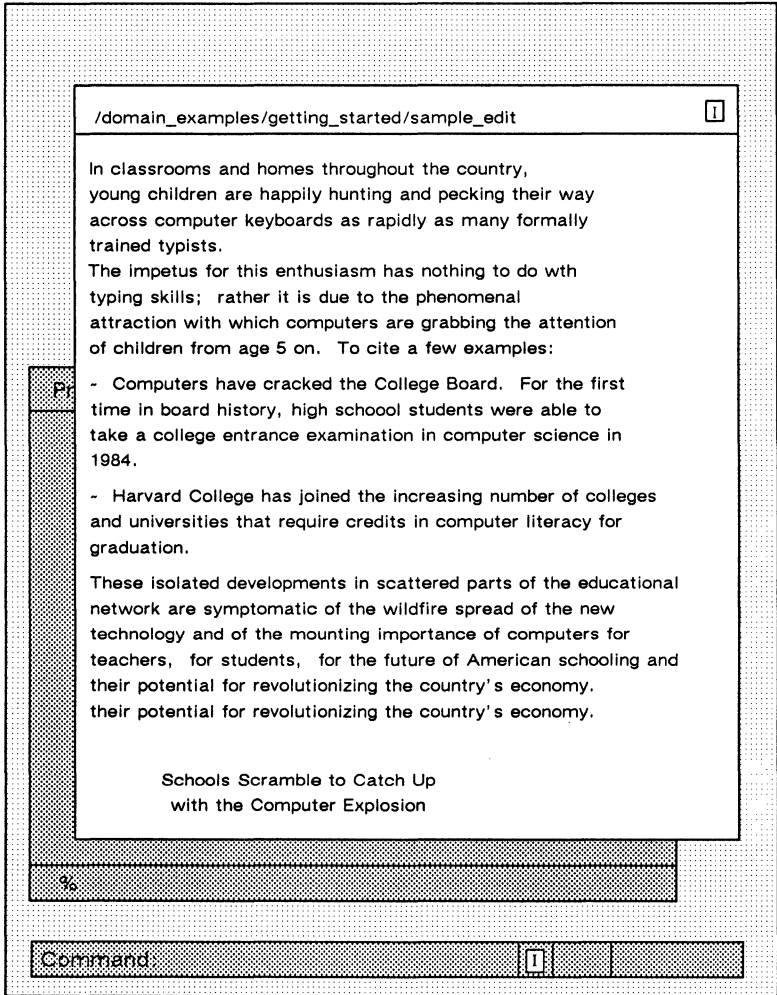


Figure 5-2. A Sample Editing Session Using the DM

Correcting Errors

Move the cursor to any text position using the arrow keys, the touchpad, or the mouse. Once you've positioned the cursor, use <BACKSPACE>, <INS>, <CHAR DEL>, and <LINE DEL> to correct errors.

Let's start correcting the errors in Figure 5-2. We omitted the letter "i" from the word "with" in the first line of the second paragraph. Position the cursor on the letter "t" and type "i". Because the edit window is in insert mode, the letters to the right of the cursor move right when you insert a new character. By default, insert mode is on when you begin an editing session, and the letter "I" appears in the edit pad's window legend.

Press <INS> to turn insert mode off. Notice that the letter "I" disappears from the window legend. Now any new characters that you type overwrite (replace) existing characters. Try this by moving the cursor to the "s" in "schooling" in the last paragraph. Type the word "education". To turn insert mode back on, press <INS> again.

Move the cursor to an "o" in "school" in the second line of the third paragraph. To delete the extra "o", press <CHAR DEL>.

To delete the extra line in the last paragraph, move the cursor onto the line and press <LINE DEL>.

Selecting Text

Before cutting, copying, or substituting text, you must select the text on which you want the DM to operate. Selecting text is also referred to as defining a range (or block) of text.

To select text, place the cursor at the beginning of the range and press <MARK>. Then move the cursor to the end of the range. As you move the cursor, the DM highlights the text, showing exactly what the range is. (Please note that the character under the cursor at the end of the range is not included within the range.) After positioning the cursor at the end of the range, either press a DM function key or type a command in the DM input window.

If you do not select text, the default range for cut, copy, and substitute commands begins at the current cursor position and extends to the end of the line.

To cancel a selection, use CTRL/X. The DM removes the highlighting that indicates the range of text you have selected.

Copying Text

The DM lets you copy text into the paste buffer without deleting it from your file. Define a range of text before you execute the DM copy operation.

The DM uses a paste buffer during a copy. A paste buffer is a temporary file that holds text you have copied so that you can paste it elsewhere.

To copy text follow these steps:

1. Select the text you wish to copy. Move the cursor to the position you wish to begin the copy and press <MARK>.
2. When you've highlighted all the text you want to copy, press <COPY>. The DM copies the text in the specified range and writes it into the paste buffer.

Cutting Text

The DM lets you delete portions of text from a file. After text is cut, you can reinsert it anywhere in the file (or even in a different file).

The DM uses a paste buffer during cut operations. A paste buffer holds text you have cut or copied so that it can be pasted elsewhere.

To cut text, follow these steps:

1. Select the text you wish to cut. Move the cursor to the position you wish to begin the cut and press <MARK>.
2. When you've highlighted all the text you want to cut, press <CUT>. (Remember to hold <SHIFT> down at the same time). The DM deletes the text in the specified range and writes it into the paste buffer.

If you accidentally cut text, restore it by pressing <PASTE> before you copy or cut anything else.

Pasting Text

To paste the contents of the paste buffer, move the cursor to where you want to paste the text. Press <PASTE>. The DM pastes the contents of the paste buffer starting at the cursor position.

Moving Text

Let's try a cut and paste operation. We'll move "Schools Scramble To Catch Up with the Computer Explosion" in Figure 5-2 from the bottom to the top of the file. Follow these steps:

1. To select the text for the operation, move the cursor to the first character position in the line before "Schools Scramble To Catch Up," and press <MARK>. Now move the cursor to the bottom of the file. As you move the cursor, notice the highlighting. This lets you see exactly what text falls within the range.
2. Press <CUT>. The DM deletes the text in the specified range and writes it into the paste buffer.

NOTE: The DM writes all copied and deleted text into this buffer; however, it saves only the text from the last DM operation. If you wish to paste the contents of the buffer, do so before performing another cut or copy.

3. To place the contents of the paste buffer at the top of the file, press CTRL/T to move the cursor to the top of the file, and then press <PASTE>.

In this exercise, you used the DM's default paste buffer to hold the text you cut. You may create your own paste buffers (up to 100), each containing different blocks of text. To learn how to create and use multiple paste buffers, see the discussion of the `xc`, `xd`, and `xp` commands in the "Editing a Pad" chapter in the user's guide for your environment.

Copying and Pasting Between Files

Now, we'll copy a range of text into a new file. Follow these steps:

1. Define any range of text in the file `sample_edit` in Figure 5-2. Move the cursor to the beginning of the range you want to copy, and press <MARK>. Then move the cursor to the end of the range. The DM highlights the text as you move the cursor.
2. Press <COPY>. The DM copies the range of text into the paste buffer. The original text remains undisturbed.
3. To create the new file, press <EDIT>. At the edit prompt in the DM input window, type `copied_text` as follows:

```
Edit file: copied_text
```

The DM opens an edit window and pad to the new file `copied_text`. The cursor is in the upper-left corner of the new file.

4. Press <PASTE>. The text copied to the paste buffer appears in the file `copied_text`.

To close `copied_text` and save its contents, keep the cursor in the window and press <EXIT>. The file is saved in your working directory.

Searching for Text

The DM search function makes it easy for you to locate a particular text string in a file. In the DM input window, you type the string between slashes (*/string/*) or backslashes (*\string*). A string enclosed in slashes instructs the DM to search forward from the current cursor position, and a string enclosed in backslashes instructs the DM to search backward from the current cursor position. For example, to search for the string “computer” in `sample_edit`, follow these steps:

1. Press CTRL/T to move the cursor to the top of the file.
2. Press <CMD> to move the cursor to the DM input window.
3. Enter the search command as follows:

Command: `/computer/`

The DM moves the cursor to the first occurrence of “computer” (in the first paragraph).

Because the DM saves your most recent search command, you may repeat a search even if you’ve entered other (non-search) commands since you entered the search command.

If you are a UNIX user, the CTRL/N and CTRL/P sequences search for the next and previous occurrence (i.e., forward and backward, respectively). For example, if you press CTRL/N now, the DM searches forward for the next occurrence of “computer” (in the second paragraph). If you then press CTRL/P, the DM moves the cursor back to the previous occurrence of computer, in the first paragraph.

For Aegis users, the repeat-search keys are CTRL/R (forward) and CTRL/U (backward).

Searches that the DM performs are case-insensitive by default. This means that `/mary/` locates `mary`, `MARY`, `Mary`, and even `mARY`. Use the DM command `sc` (set case) to enable case-sensitive searching. For more information on the `sc` command, see the chapter “Editing a Pad” in the user’s guide for your environment.

Canceling a Search

To cancel a search operation, use CTRL/X. The DM displays the message "Search aborted".

Substituting Text

The DM also provides an easy-to-use search and substitute command, *s* (substitute). Use it to search a file or part of a file for a text string, and to replace the string with a new string.

For example, to replace every tilde (~) in `sample_edit` with the letter "o", you must

1. Select the text you wish to search. The range for this search/substitute operation is from the top to the bottom of the file. To select the entire file,
 - A. Move the cursor to the top of the file (CTRL/T does this) and press <MARK>.
 - B. Next move the cursor to the bottom of the file (CTRL/B does this) and press <CMD>.
2. Next, enter the following substitute command:

Command: `s/~o/`

The DM replaces every tilde (~) in the file with the letter "o".

When you do not mark a range, the DM searches only the line to the right of the cursor when you press <CMD> in Step 1B.

Undoing Previous Commands

The DM keeps a history of activities in the DM input pad and edit pads in reverse chronological order. The undo function, invoked with <UNDO>, makes use of this information to reverse the effect of the most recent DM command. Successive use of the UNDO key undoes DM commands further back in history.

The history of DM activities consists of circular lists (one for input pad activities and one for each edit pad). When the lists are full, they eliminate the oldest entries to make room for new ones.

NOTE: <UNDO> works for DM operations only; it does not reverse the effect of shell commands once they are executed.

Saving an Edit File

To save a file, move the cursor into the edit window and press <SAVE>. The DM saves your file and changes the window to read-only mode. To make additional changes, press SHIFT/<AGAIN> first.

Closing an Edit File

You can save an edit file when closing its window, or you can discard the file without saving it.

Closing and Saving a File

To save a file and close the window in one operation, leave the cursor in the window and press <EXIT>. Doing so also renames the original version (if any) of the file for backup purposes. This backup copy of the file appears in your directory with the original filename plus the .bak suffix.

Discarding a File

If you decide that you don't want to save the file, use <ABORT> to discard it. When you press <ABORT>, if you have made unsaved changes to the file, the DM displays the following:

File modified. OK to quit?

Enter `n` or `no` to resume editing. To discard the changes you've made, enter `y` or `yes`; the DM closes the edit pad and window without saving the changes you made during the session.

Printing a File

The command for printing files is `prf`. You can use this command in two ways: to call up a visual aid (a dialogue box), or as an immediate print command. The visual aid, or print menu, helps you learn what arguments and options you want. Once you are comfortable with the options for `prf`, you may prefer using the command without the dialogue option, as this method is faster.

With the Dialogue Option

If you use the dialogue option a print menu appears (see Figure 5-3). This menu lets you specify various arguments and options by typing information and making selections from the menu.

To display the print menu on your screen, enter the shell command as follows:

```
% prf -dia
```

NOTE: This example shows the command as it would appear when entered at a C shell prompt. The command line will work no matter which environment you have.

The `prf` command displays the print menu in a special window at the top of the shell process transcript pad. An arrow cursor appears in the upper left corner of the menu.

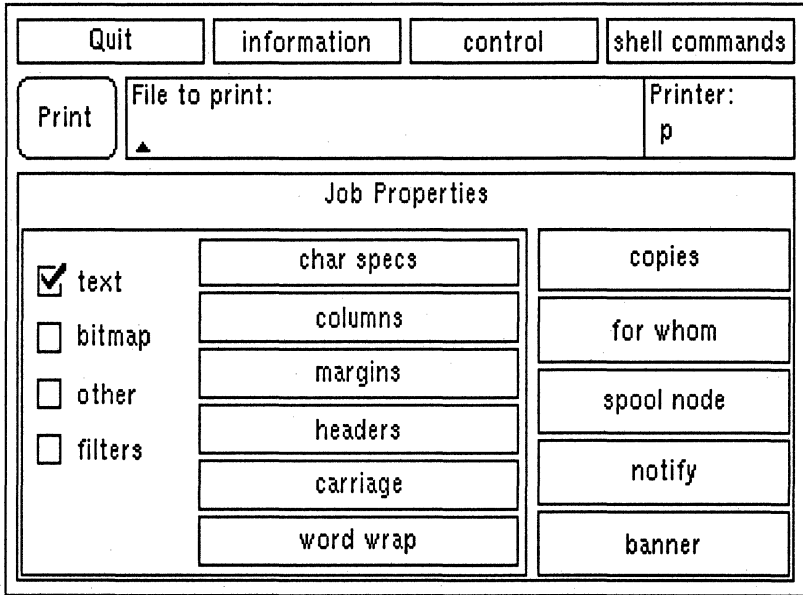


Figure 5-3. The Print Menu

The triangular cursor below the “File to print:” prompt indicates that characters typed at the keyboard will appear in this field. To print the file `copied_file` in your working directory, first you must type its name followed by `<RETURN>`:

```
File to print:
[copied_file ]
```

Note the square brackets which enclose the filename as you type. When present, these indicate that you have not yet pressed `<RETURN>`.

If you want to print a file from a directory other than the current one, enter a full pathname like `//node/owner/copied_file`.

To choose a printer other than the default for your node, move the arrow cursor to the space below the “Printer:” prompt. Activate the typing cursor by pressing <F1> or by clicking the left mouse button, and type the name of the printer you want to use.

The number of copies to print is set to one initially. To change this value, select the “copies” field with the left mouse button or with <F1>, and enter the desired number of copies.

For help information on any of the print menu fields, position the cursor over the field and press the HELP key (in the upper right-hand corner of your keyboard); be sure to hold down <SHIFT> first. If you have a mouse, clicking the right mouse button will also display help information.

When you’re satisfied with the information that you’ve supplied on the print menu, move the arrow cursor to the box labeled Print. Press function key <F1> or, if you are using a mouse, press the left mouse key. A message similar to the following appears on the transcript pad:

```
//node/owner/copied_file queued to printer lq
```

The print menu remains on your screen, allowing you to print additional files. When you’re ready to exit from the print menu, move the arrow cursor to the box labeled “Quit” and press <F1> or the left mouse key. The menu disappears from your screen, and the cursor reappears at the shell prompt.

Without the Dialogue Option

To print a file immediately using the `prf` command, enter a command of the form

```
% prf pathname
```

To use a printer other than the default for your node, use the `-pr` option followed by the name of the printer:

```
% prf copied_file -pr spin
```

NOTE: To find out the names of the printing devices at your site, type

```
% prf -list_printers
```

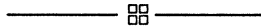
When you give the **prf** command shown above, the program confirms that your file has been sent to the printer by displaying a message like

```
//node/owner/copied_file queued to printer spin
```

The **prf** command has a number of other options. For more information, see the “Managing Files” chapter in the user’s guide for your environment.

Summary

In this chapter, you learned how to create, edit, read, and print files.



Part II

The SysV Environment

Chapter 6

Introducing SysV

The SysV environment is an implementation of the UNIX operating system (AT&T System V, release 3).

This chapter introduces the SysV environment. All shell examples in Chapters 6–8 use the Bourne shell, the standard shell used with AT&T System V.

What Is a Shell?

The shell program provides access to traditional computing operations, such as printing documents, compiling and running programs, and monitoring system activities. The shell “listens” for commands typed at your keyboard. Shell commands invoke **utilities**, which are programs that perform tasks you request.

Using UNIX Commands

When you press <SHELL>, the DM creates a new process running a Bourne shell program and displays the windows associated with the new shell process.

Move the cursor to the Bourne shell prompt in the process input window and enter the `date` command. Enter the command exactly as shown (i.e., in lowercase letters).

```
$ date
```

In the process output window, the shell displays the command itself and then invokes the utility program that displays the date and time. For example, the output of the command you just typed might be

```
Thu Jul 21 14:49:04 EDT 1988
```

Getting Help

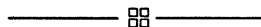
To get information about available UNIX commands, library and system calls, and functions, use the `man` command. This command lets you select and display online versions of reference material from the *SysV Command Reference* and the *SysV Programmer's Reference*. For example, to display the manual page for the `ls` command, type the following:

```
$ man ls
```

The `man` command opens a separate read window containing a formatted version of the manual page(s) for the `ls` command. While the manual page is displayed, you may continue to execute other UNIX shell commands (including more `man` commands). When you finish reading the manual page, use `<EXIT>` to close the window.

Summary

In the next chapter, you'll learn about commands that manipulate files and directories.



Chapter 7

Working with Files and Directories in SysV

This chapter describes how to use directories, and how to copy and delete files. It also mentions two other available editors, `vi` (a display-oriented editor) and `ed` (a line editor).

Working with Directories

This section explains some of the most common ways you'll work with directories.

Listing the Contents of a Directory

Use the `ls` (list directory) command to list the contents of a directory. To display the contents of your network root directory, type the `ls` (list directory) command and the pathname `//` as follows:

```
$ ls //
```

Use the `ls` command to list the contents of any directory in your network. You can refer to any object in your network by specifying a pathname that begins with the root directory (`//`). You must have the appropriate permissions to access the objects you specify.

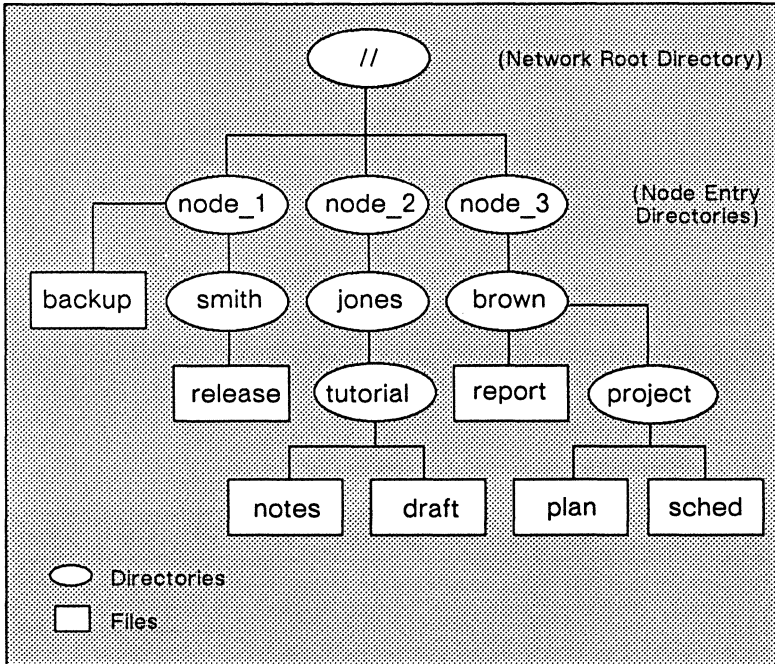


Figure 7-1. A Sample Naming Tree

Listing the Contents of Your Node Entry Directory

Use a single slash (`/`) as a shorthand way of referring to your node's entry directory. For example, the following command line lists the contents of your node entry directory:

```
$ ls /
```

When you begin a pathname with a single slash (/), the system begins its search in the node entry directory of the node that is physically connected to your display unit. (If your node is diskless, the system begins its search in the entry directory of your node's disked partner.)

You may also list the contents of your node entry directory by typing the `ls` (list directory) command and a pathname that includes the root directory (//) and your node entry directory name. For example, if you are working on `node_1` in the sample naming tree shown in Figure 7-1, the following command lists `backup` and `smith`:

```
$ ls //node_1
```

To refer to the entry directory on another node, use a pathname that starts with the root directory (//) and includes the name of the other node's entry directory.

Listing the Contents of Your Working Directory

To list the contents of your working directory, type

```
$ ls .
```

or

```
$ ls
```

Listing the Contents of Your Parent Directory

To display the parent directory's contents, type

```
$ ls ..
```

The double periods (..) refer to the directory one level above the working directory. Use a combination of double periods and slashes to back up more than one level above the working directory.

For example, let's say that your working directory is `//node_1/smith/newdir/dir1/data`, and you want to list the directory `//node_1/smith/newdir`. Type the following:

```
$ ls ../../
```

Your Home Directory

You can refer to your home directory with the tilde symbol (`~`). Thus, no matter which directory you are working in, you can display the contents of your home directory with the command

```
$ ls ~
```

You can also use this symbol (the tilde) as the start of a longer pathname. For example, if your home directory contains a subdirectory `programs`, you can find out its contents from anywhere in the naming tree by typing

```
$ ls ~/programs
```

Changing Directories

Change to another directory using the `cd` (change directory) command. For example,

```
$ cd //node_3/brown/project
```

changes your working directory to `//node_3/brown/project`.

You can go from any location in the network naming tree to your home directory by using the `cd` command without any arguments:

```
$ cd
```

Identifying Your Working Directory

To display the name of the directory you're using (your working directory), enter the `pwd` (print working directory) command as follows:

```
$ pwd
```

The shell prints the name of your working directory.

Creating a New Directory

Use the `mkdir` (make directory) command to create a new directory in your working directory. For example, if your working directory is `//node_1/smith`, and you wish to create a new directory named `newdir`, type

```
$ mkdir newdir
```

The new directory, `//node_1/smith/newdir` becomes a subdirectory of the working directory, `//node_1/smith`. The new directory appears beneath the working directory in the naming tree. If you enter the `ls` command, `newdir` appears in the list of the working directory contents. Figure 7-2 illustrates the location of `newdir` in the naming tree.

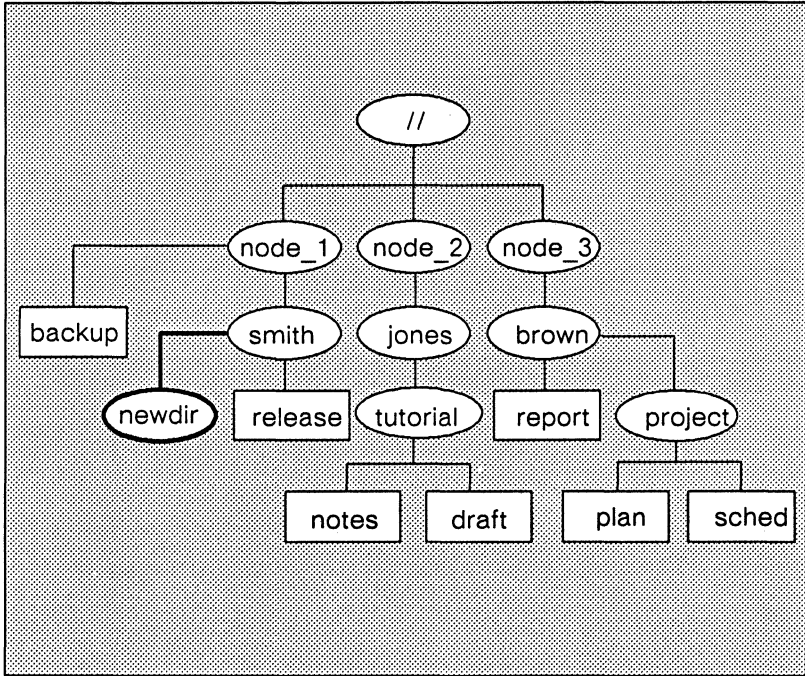


Figure 7-2. A New Directory in the Naming Tree

Pathname Symbols

You may refer to any object in your network by using a pathname that begins with the root directory (`//`). The system also provides shorthand symbols to use in place of longer pathnames. Table 7-1 summarizes pathname starting-point symbols.

Table 7-1. Pathname Starting-point Symbols

If your pathname starts with this symbol:	The system begins the name search in this directory:
//	Network root directory
/	Node entry directory
No symbol or	Working directory
..	Parent directory
~	Home directory

Figure 7-3 illustrates how to use different symbols to refer to the same directory in the naming tree.

Remember that the examples in Figure 7-3 assume that the node entry directory is `//node_x` and that the working directory is `//node_x/john`. Each of the pathnames in Figure 7-3 searches for a destination object named `mary`.

The object's full pathname is `//node_x/mary`. The pathname `/mary` instructs the system to look for `mary` in the node's entry directory (`/`). The pathname `../mary` instructs the system to move up to `//node_x` and then look for an object named `mary`.

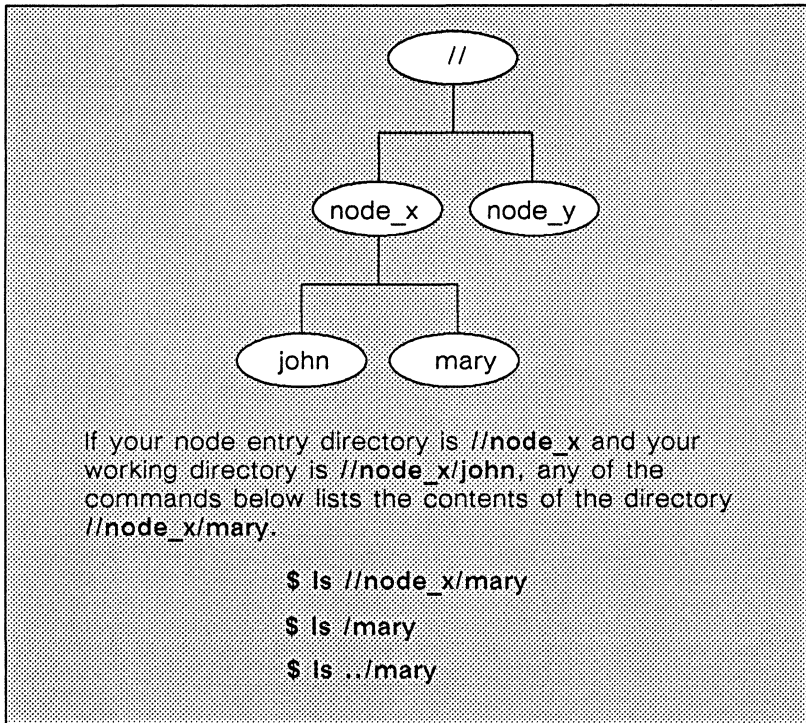


Figure 7-3. Pathnames Starting with `//`, `/`, and `../`

Working with Files

This section explains how to copy and delete files.

Copying a File

Use the `cp` (copy) command to copy the contents of a file and store it in another file. Specify both the name of the file to be copied and the name to give the new file. For example, to make a copy of `sample_edit` and store it in a new file called `copied_file` in your working directory, enter the following command:

```
$ cp sample_edit copied_file
```

If you enter the `ls` command at the UNIX shell prompt, you'll see `copied_file` listed in the contents of your working directory.

Deleting a File

To delete files from directories, use the `rm` (remove) command with the name of the file you want removed. The following example deletes the file `copied_file` from your working directory:

```
$ rm copied_file
```

To delete a file anywhere else in the network, supply a pathname. You must, however, have write permission for the file before you can delete it.

Using Links

As you use the system, you may frequently want to access objects with very long pathnames. Instead of typing the long pathname each time you want to use an object, you can create a special object type called a link. A link gives you a shortcut from one part of the naming tree to another.

You can create links of two types: hard links and soft links. **Hard** links point directly to a file, and they may not span file systems (disks) or refer to directories. **Soft** links, however, contain link text (an object's pathname), can span file systems, and may refer to directories. Although you may want to use soft links more frequently, we'll discuss both types of links in this section.

Creating a Soft Link

Let's say that you work on `node_1` in our sample naming tree (Figure 7-1), and you often use the file `sched`. To access `sched`, you could type the pathname `//node_3/brown/project/sched` each time you want to use the file.

It is much easier, however, to create a link to `sched`. Then, whenever you want to access `sched`, you can use the name of the link to get to it. To create a soft link, use the `ln` (link) command with the `-s` option. Let's call the link that represents the `sched` file in our example `mylink`. To create `mylink` and place it in your working directory, you would type

```
$ ln -s //node_3/brown/project/sched mylink
```

This creates a soft link. A soft link contains the name of the file to which it is linked (the `link text`). Thus, when you use a link name as a pathname or as part of a pathname, the link text (in this case, `//node_3/brown/project/sched`) is substituted for the link name (`mylink`).

NOTE: Soft links are a feature of Domain/OS that Apollo has added to System V.

Creating a Hard Link

By default (with no options specified), the `ln` command creates hard links. Thus, if you type the command line

```
$ ln //node_3/brown/project/sched mylink
```

The link `mylink` points directly to `//node_3/brown/project/sched`. Hard links are indistinguishable from the original directory entry to which they point, so any changes made to the file are independent of the name used to reference the file.

Therefore, if you create a hard link to `sched` and save a new copy of the file using the DM editing function, your link points to the original file (called `sched.bak` after the editing session is complete) and not to the newly modified version of the file (called `sched` after the editing session is complete).

To have your link always pointing to the latest version of `sched`, create a soft link using `ln -s` as shown above.

For more information about links, see the `ln` command description in the *SysV Command Reference*, or refer to the chapter "Managing Links" in *Using Your SysV Environment*.

The UNIX Display-Oriented Editor (vi)

You may prefer to use `vi`, the UNIX display-oriented (visual) text editor, when editing files. When you use `vi`, the position of the cursor on the screen indicates the position within the file. The editor uses commands that let you manipulate text in a variety of ways.

Consult the *UNIX Text Processing* manual for complete details on the use of `vi`. For a much briefer description of usage see the *SysV Command Reference*. To view an online display of the command reference page for `vi`, type the following:

```
$ man vi
```

The UNIX Line Editor (ed)

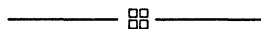
The standard UNIX line editor, `ed`, allows you to edit files on a line-by-line basis. Unlike `vi`, it does not permit you to view large sections of text at once.

The *UNIX Text Processing* manual and the *SysV Command Reference* both provide more information. To view an online display of the command reference page for `ed`, type the following:

```
$ man ed
```

Summary

In this chapter, you learned how to copy and delete files. You also learned how to use directories and links, and about the additional editors available under the SysV environment. The next chapter contains specific information about the Bourne shell.



Chapter 8

Using the Bourne Shell

This chapter explains how to use the Bourne shell, the default SysV shell. In this chapter, you'll learn about wildcards, symbols that redirect input and output, pipes and filters, and shell scripts.

Command Format

The simplest UNIX shell command consists of a command name alone, for example,

```
$ ls
```

The `ls` (list directory) command lists the contents of your working directory.

Using Command Arguments

Most shell commands accept one or more **command arguments** to specify the object (file, directory, or link) upon which the command acts, as in the case below:

```
$ ls your_directory
```


In this example, `your_directory` is the argument. The `ls` command lists the contents of `your_directory`. You must use one or more spaces to separate a command from its arguments, and to separate arguments from each other.

Using Command Options

In addition to command arguments, most shell commands accept **command options** to modify the action of commands. They appear before command arguments. Most options are preceded by a hyphen, although this is not always the case. Check the *SysV Command Reference*, or the online manual page (using the `man` command), for specific usage instructions in the case of each option. If an option is preceded by a hyphen, include a space before the hyphen (but not between the hyphen and the option), as in the case

```
$ ls -l
```

The `ls` command, used with the `-l` option displays the contents of the working directory in “long” format, supplying permissions, number of links, owner, size in bytes, and time of last modification for each file in the directory.

Entering Multiple Commands on a Line

To enter more than one command on a single line, separate the commands with a semicolon (;). For example:

```
$ ls; date
```

This command lists the contents of the working directory, then displays the date. Here’s another example you may want to try:

```
$ cd; ls
```

The `cd` (change directory) command changes the working directory to your home directory, and then the `ls` command lists its contents.

Command Line Processing

Most commands are the names of files. When you press <RETURN> to enter one of these commands, the shell searches for a file with the same name as the command you specified. If the shell finds such a file, it loads and executes it as a program.

Next, the shell passes any arguments and options on your command line to the program with the same name as the command. After executing the program, the shell displays a new prompt to show that it is ready for your next command. For example, when you enter the command

```
$ ls -l my_directory
```

the shell searches for a file named `ls`. The shell loads the program, passes the option and the argument to it, and executes the program. After the program displays the contents of `my_directory` in a single column, the shell redisplay a prompt on its input pad. You may then enter another shell command.

Command Search Paths

When you enter a command that is the name of a file, the UNIX shell takes a particular route (referred to as a **command search path**) in determining which directories to search for the command file. You may display the current search path of the UNIX shell you're using by typing the `echo` command, with `$PATH` as its argument:

```
$ echo $PATH
```

NOTE: `PATH` is an instance of a shell variable. See *Using Your SysV Environment* for a fuller discussion of shell variables.

The following directory names are displayed in response to the `echo` command line above:

```
:/bin:/usr/bin:/usr/apollo/bin
```

Now let's take a closer look at this information. The sequence of directory names separated by colons above is stored in the shell variable called `PATH`. The shell looks for command files in this order:

1. Your working directory (`.`), implied by the empty field before the first colon
2. The main system command directory (`/bin`)
3. A second, less frequently used, system command directory (`/usr/bin`)
4. A directory containing Apollo-specific commands (`/usr/apollo/bin`)

As soon as the shell finds a filename that matches the command you specify, it attempts to execute the program.

In the Bourne shell, you can change the search path by adding new values to the `PATH` shell variable. For example, typing

```
$ PATH=$PATH:~/bin
$ export PATH
```

at the Bourne shell prompt adds the `~/bin` directory to the end of your search path. Typing the `echo` command with `$PATH` as its argument shows

```
:/bin:/usr/bin:/usr/apollo/bin:~/bin
```

Using Wildcards

Wildcards are special characters that you may use to represent one or more pathnames. For example, the following command line contains the `*` wildcard:

```
$ ls *.bak
```

The expression `*.bak` matches every file that ends in `.bak` (backup versions of text files). Therefore, this command line lists all the files that end in `.bak` in your working directory.

The next example lists all the objects in your home directory beginning with one of the letters `a` through `z`.

```
$ ls ~/[a-z]*
```

The `[...]` wildcard, where the ellipsis (...) represents specified characters, matches any of the enclosed characters. A pair of characters separated by a dash (-) matches any character lexically between the pair. Thus `[2-5]` matches 2, 3, 4, and 5, and `[A-Z]` matches any capital letter.

Do not use wildcards until you are sure that you understand fully how they operate. “Using the Bourne Shell” in *Using Your SysV Environment* gives more detailed information about wildcards.

Redirecting Input and Output

In the previous examples in this book, the shell process read the commands you entered (input) from its input pad, and it wrote the responses to your commands (output) on its transcript pad. The next two sections explain a few examples of special characters that redirect input and output. For complete information about these characters, see the chapter “Using the Bourne Shell” in *Using Your SysV Environment*.

Writing Output to a File

Some shell commands write output to the transcript pad. For example, the following `cat` (catenate) command line, with a filename as an argument, directs its output to the transcript pad:

```
$ cat myfile
```

Here, `cat` catenates `myfile` and displays the output on the transcript pad.

To direct the output to a file instead of the transcript pad, use the greater-than symbol (`>`) and another filename. For example,

```
$ cat myfile > myfile.out
```

writes the catenated text to `myfile.out`. The shell creates the output file (`myfile.out`) if it doesn't already exist. Note that if `myfile.out` already exists, the shell deletes its old contents first.

Reading Input from a File

To instruct a shell command to read input from a file, use the less-than symbol (`<`) and a filename.

In our next example, we use the `wc` (word count) command, which counts the number of characters (including blank spaces), words (character strings delimited by spaces, tabs, and newlines), and lines contained in the standard input.

Before executing the `wc` command, create a text file to use as input to the command. To do so, follow these steps:

1. Create an edit file. Name it `sample_file`.

2. When the DM creates the edit window and pad on your screen, type the following lines of text:

wc counts the number of words, characters, and lines contained in the standard input (by default) or in a file that you specify.

3. Press <EXIT> to close the edit window and pad.

Now instruct `wc` to get its input from the file `sample_file` with the following command:

```
$ wc < sample_file
```

This tells `wc` to supply a count of the lines, words, and characters contained in `sample_file`. Then `wc` writes its output to the transcript pad:

```
4      23      129
```

The output represents lines (4), words (23), and characters (129) found in `sample_file`.

Using Pipes and Filters

A **pipe**, represented by the vertical-bar character (`|`), connects the standard output of one shell command to the standard input of another. For example, suppose you want to know how many words two files contain when combined. You could discover this with the two command lines

```
$ cat file1 file2 > sample_file
$ wc -w < sample_file
```

Clearly, this method is cumbersome. And you probably don't want to keep a copy of the union of the two files; all you really need is a report on the total number of words contained in the two files. A pipe lets you do both tasks in one simple operation called a **pipe-line**:

```
$ cat file1 file2 | wc -w
```

In this case, the `cat` command creates an output file only on the standard output and not in your working directory. The `wc` command uses this temporary data as standard input to produce the results on your display.

The length of a pipeline is not restricted to a particular number of operations. However, you should remember that pipes are interpreted by the shell as being unidirectional: that is, commands to the right of the vertical-bar character cannot send data to commands to the left of it.

Although a regular pipe that has several programs feeding into one final program won't work, you can build a type of "T-joint" in the pipe by using the `tee` command in your pipe. This command tells the shell to transcribe the standard input to the standard output and make copies in the named files.

The `tee` command is particularly useful for saving data being transmitted through the pipe, data that is written over in the process. Suppose you want to check a file called `test` for spelling errors, create a file named `test.errors` that contains the list of misspelled words found in `test`, and create a separate file named `test.num` that provides only a count of the total number of misspellings in `test`. You could use the following command line:

```
$ spell test | tee test.errors | wc -w > test.num
```

Because `tee` copies its input to the specified file(s) and to its output, the intermediate output from the execution of `spell` is saved in `test.errors`. A copy of this data passes through the `wc` program, and the result of the execution of `wc` is written into a new file named `test.num`.

A **filter** reads its input, performs some specified operation on it, and then prints the result as output. One useful filter, **fgrep**, searches its input for lines that contain a specified string. For instance,

```
$ ls | fgrep bak
```

prints all lines from the output of **ls** that contain the string “bak”.

Creating Shell Scripts

A **shell script** is a file that contains a list of shell commands. If you find yourself repeating a sequence of commands over and over again, create a shell script containing the commands. Then you can execute the entire command sequence with a single command.

To create a script, use **<EDIT>** to create the file; type the shell commands, one per line, then press **<EXIT>** to close the file. To execute the script, type the script’s filename in the process input window next to the UNIX shell prompt. Be sure to read the remainder of this section for important details about shell script execution.

Let’s try an example. Suppose you want to change your current directory to your home directory, and then get a long listing of its contents (using the **-l** option with **ls**). You could accomplish these tasks by entering these shell commands, one after another:

```
$ cd
$ ls -l
```

The **cd** (change directory) command sets the working directory to your home directory. The **ls** command displays the name of each object in your home directory. The **-l** option displays each object’s permissions, owner, size in bytes, etc.

If you create a shell script containing the `cd` and `ls -l` command lines, you may execute both commands by simply entering the filename of the script. To create a script called `dir`, follow these steps:

1. Go to your home directory by entering the command

```
$ cd
```

2. Create a personal directory in which your shell scripts can reside. Typically, your built-in scripts should reside in a `bin` subdirectory of your home directory. You already have a `/bin` subdirectory in your node entry directory, and the UNIX shell uses this subdirectory in its command search path. Add your personal `bin` to your search path so that the shell looks there for commands as well. Go to your `bin` subdirectory by typing

```
$ cd bin
```

NOTE: Now, if you type `pwd` at the UNIX shell prompt, you should see your home directory followed by the name `bin`.

3. Create an edit file named `dir`.
4. When the DM creates the edit window and pad, type the shell commands, one per line, like this:

```
cd my_dir  
ls -l
```

5. Press `<EXIT>` to close the edit window and pad.

Because files created with the DM editor are by default not executable, you must also enter the command

```
$ chmod +x dir
```

NOTE: The command `chmod` (change mode) is used to modify an object's permissions. For more information, see the chapter "Controlling Access to Files and Directories" in *Using Your SysV Environment*.

Using the script that you created, you can now execute the `cd` and `ls` command lines by typing

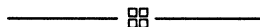
```
$ dir
```

Remember that once you have included your personal `bin` directory in your search path, you may execute your shell scripts from anywhere in the naming tree. When you're ready to create more sophisticated scripts, read "Using the Bourne Shell" in *Using Your SysV Environment*.

Summary

In this chapter, you learned more about UNIX shells. You read about how a shell processes your commands and how it locates the command files you request. We also introduced you to some of the more powerful features of UNIX shells, namely wildcards, symbols that redirect input and output, pipes and filters, and shell scripts.

Now that you're familiar with the UNIX shells, read *Using Your SysV Environment* to learn how to perform specific tasks using either the Bourne shell or the C shell. See the *SysV Command Reference* for a brief explanation of how each shell works, or use `man` to view information on the C shell (`csh`), the Bourne shell (`sh`), or the Korn shell (`ksh`).



Part III

The BSD Environment

Chapter 9

Introducing BSD

The BSD environment is an implementation of the UNIX operating system (4.3 BSD from the University of California at Berkeley). All examples in Chapters 9–11 use the C shell, the standard shell for BSD.

What Is a Shell?

The shell program provides access to traditional computing operations, such as printing documents, compiling and running programs, and monitoring system activities. The shell “listens” for commands typed at your keyboard. Shell commands invoke **utilities**, which are programs that perform tasks you request.

Using UNIX Commands

When you press <SHELL>, the DM creates a new process running a shell program. The kind of shell (Bourne, C, or Korn) depends on your system’s configuration. Since we use the C shell for all

examples in Part III, you may want to create shells by entering the following command in the DM input window:

Command: `cp /bin/csh`

Move the cursor next to the C shell prompt in the process input window and enter the `date` command. Enter the command exactly as shown (i.e., in lowercase letters).

`% date`

In the process output window, the shell displays the command itself and then invokes the utility program that displays the date and time. For example, the output of the command you just typed might be

```
Thurs Feb 4 16:54:24 EDT 1988
```

Getting Help

To get information about available UNIX commands, library and system calls, use the `man` command. This command lets you select and display online versions of reference material from the *BSD Command Reference* and the *BSD Programmer's Reference*. For example, to display the manual page for the `ls` command, type the following:

`% man ls`

The `man` command opens a separate read window containing a formatted version of the manual page(s) for the `ls` command. While the manual page is displayed, you may continue to execute other UNIX shell commands (including more `man` commands). When you finish reading the manual page, use `<EXIT>` to close the window.

Summary

After reading this chapter, you should know how to enter UNIX commands and get help. In the next chapter, you'll learn about commands that manipulate files and directories.



Chapter 10

Working with Files and Directories in BSD

This chapter describes how to use directories, and how to copy and delete files. It also mentions two other available editors, `vi` (a display-oriented editor) and `ed` (a line editor).

Working with Directories

This section explains some of the most common ways you'll work with directories.

Listing the Contents of a Directory

Use the `ls` (list directory) command to list the contents of a directory. To display the contents of your network root directory, type the `ls` (list directory) command and the pathname `//` as follows:

```
% ls //
```

Use the `ls` command to list the contents of any directory in your network. You can refer to any object in your network by specifying a pathname that begins with the root directory (`//`). You must have the appropriate permissions to access the objects you specify.

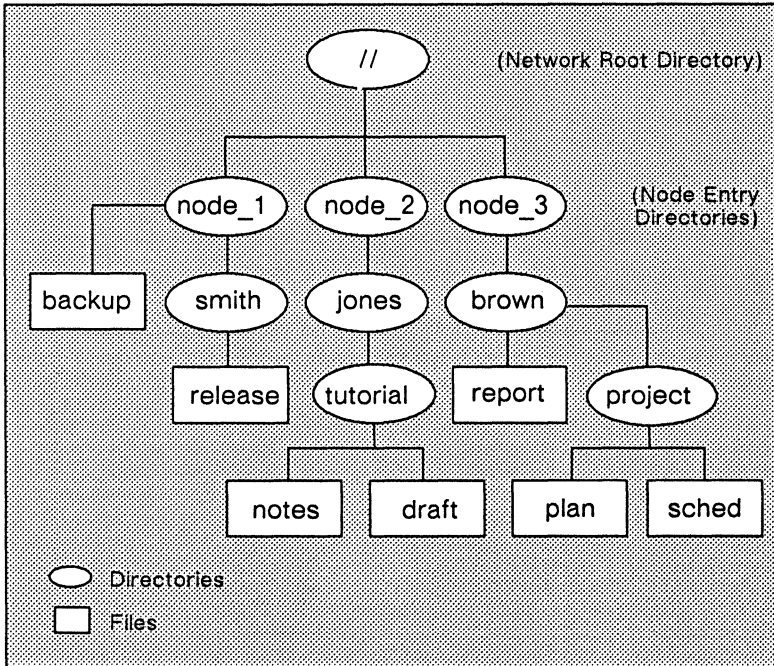


Figure 10-1. A Sample Naming Tree

Listing the Contents of Your Node Entry Directory

Use a single slash (`/`) as a shorthand way of referring to your node's entry directory. For example, the following command line lists the contents of your node entry directory:

```
% ls /
```

When you begin a pathname with a single slash (/), the system begins its search in the node entry directory of the node that is physically connected to your display unit. (If your node is diskless, the system begins its search in the entry directory of your node's disked partner.)

You may also list the contents of your node entry directory by typing the `ls` (list directory) command and a pathname that includes the root directory (//) and your node entry directory name. For example, if you are working on `node_1` in the sample naming tree shown in Figure 10-1, the command

```
% ls //node_1
```

lists `backup` and `smith`.

To refer to the entry directory on another node, use a pathname that starts with the root directory (//) and includes the name of the other node's entry directory.

Listing the Contents of Your Working Directory

To list the contents of your working directory, type

```
% ls .
```

or

```
% ls
```

Listing the Contents of Your Parent Directory

To display the parent directory's contents, type

```
% ls ..
```

The double periods (..) refer to the directory one level above the working directory. Use a combination of double periods and slashes to back up more than one level above the working directory.

For example, let's say that your working directory is `//node_1/smith/newdir/dir1/data`, and you want to change the working directory to `//node_1/smith/newdir`. Type the following:

```
% cd ../../
```

Your Home Directory

You can refer to your home directory with the tilde symbol (`~`). Thus, no matter which directory you are working in, you can display the contents of your home directory with the command

```
% ls ~
```

You can also use this notation as the start of a longer pathname. For example, if your home directory contains a subdirectory `programs`, you can find out its contents from anywhere in the naming tree by typing

```
% ls ~/programs
```

Changing Directories

Change to another directory using the `cd` (change directory) command. For example,

```
% cd //node_3/brown/project
```

changes your working directory to `//node_3/brown/project`. You can go from any location in the network naming tree to your home directory by using the `cd` command by itself:

```
% cd
```

Identifying Your Working Directory

To display the name of the directory you're using (your working directory), enter the `pwd` (print working directory) command as follows:

```
% pwd
```

Creating a New Directory

Use the `mkdir` (make directory) command to create a new directory. The new directory will be created in your working directory. For example, if your working directory is `//node_1/smith`, and you wish to create a new directory named `newdir`, type:

```
% mkdir newdir
```

The new directory, `//node_1/smith/newdir` becomes a subdirectory of the working directory, `//node_1/smith`. The new directory appears beneath the working directory in the naming tree. (If you enter the `ls` command, `newdir` appears in the list of the working directory contents.) Figure 10-2 illustrates the location of `newdir` in the naming tree.

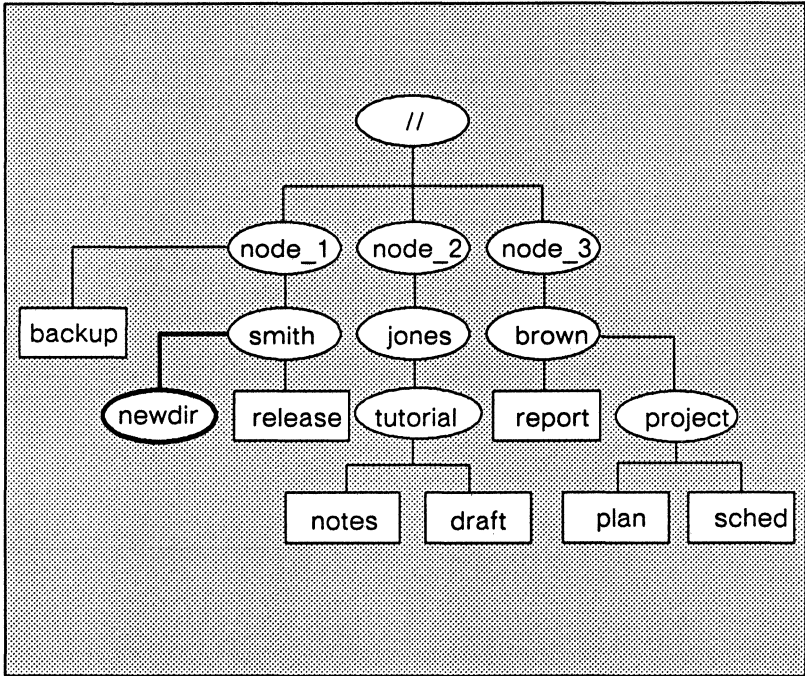


Figure 10-2. A New Directory in the Naming Tree

Pathname Symbols

You may refer to any object in your network by using a pathname that begins with the root directory (`//`). The system also provides shorthand symbols to use in place of longer pathnames. Table 10-1 summarizes pathname starting-point symbols.

Table 10-1. Pathname Starting-point Symbols

If your pathname starts with this symbol:	The system begins the name search in this directory:
//	Network root directory
/	Node entry directory
No symbol or .	Working directory
..	Parent directory
~	Home directory

Figure 10-3 illustrates how to use different symbols to refer to the same directory in the naming tree.

Remember that the examples in Figure 10-3 assume that the node entry directory is `//node_x` and that the working directory is `//node_x/john`. Each of the pathnames in Figure 10-3 searches for a destination object named `mary`.

The object's full pathname is `//node_x/mary`. The pathname `/mary` instructs the system to look for `mary` in the node's entry directory (`/`). The pathname `../mary` instructs the system to move up to `//node_x` and then look for an object named `mary`.

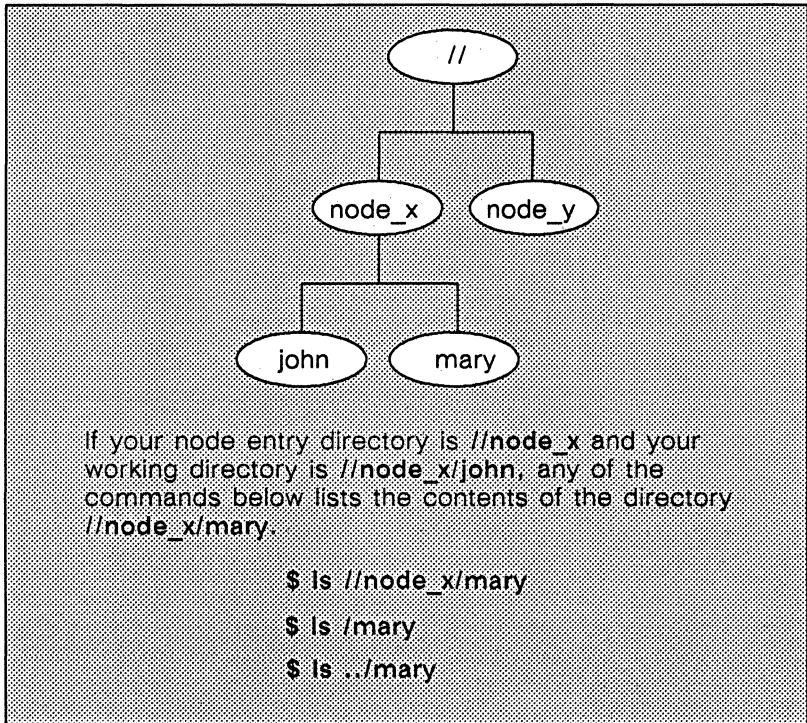


Figure 10-3. Pathnames Starting with `//`, `/`, and `../`

Working with Files

This section explains how to copy and delete files.

Copying a File

Use the `cp` (copy) command to copy the contents of a file to another file. Specify both the name of the file to be copied and the name to give the new file. For example, to make a copy of `sample_edit` and store it in a new file called `copied_file` in your working directory, enter the following command:

```
% cp sample_edit copied_file
```


If you enter the `ls` command at the UNIX shell prompt, you'll see `copied_file` listed in the contents of your working directory.

Deleting a File

To delete files from directories, use the `rm` (remove) command with the name of the file to be removed. For example,

```
% rm copied_file
```

deletes the file `copied_file` from your working directory. To delete a file elsewhere in the network, supply a pathname. You must, however, have write permission for the file before you can delete it.

Using Links

As you use the system, you may frequently want to access objects with very long pathnames. Instead of typing the long pathname each time you want to use an object, you can create a special object type called a link. A link gives you a shortcut from one part of the naming tree to another.

Links are essentially of two types: hard links and soft links. **Hard** links point directly to a file, and they may not span file systems (disks) or refer to directories. **Soft** links, however, contain link text (an object's pathname), can span file systems, and may refer to directories. Although you may want to use soft links more frequently, we'll discuss both types of links in this section.

Creating a Soft Link

Let's say that you work on `node_1` in our sample naming tree (Figure 10-1), and you often use the file `sched`. To access `sched`, you could type the pathname `//node_3/brown/project/sched` each time you want to use the file.

However, it is much easier to create a link to `sched`. Then, whenever you want to access `sched`, you can use the name of the link to get to it. To create a soft link, use the `ln` (link) command with the `-s` option. Let's call the link that represents the `sched` file in our example `mylink`. To create `mylink` and place it in your working directory, type the following:

```
% ln -s //node_3/brown/project/sched mylink
```

This creates a soft link. A soft link contains the name of the object to which it is linked (the link text). Thus, when you use a link name as a pathname or as part of a pathname, the link text (`//node_3/brown/project/sched`) is substituted for the link name (`mylink`).

Creating a Hard Link

By default (with no options specified), the `ln` command creates hard links. Thus, if you type the following command line,

```
% ln //node_3/brown/project/sched mylink
```

`mylink` points directly to `//node_3/brown/project/sched`. Hard links are indistinguishable from the original file to which they point.

Therefore, if you create a hard link to `sched` and save a new copy of the file using the DM editing function, your link points to the original file (called `sched.bak` after the editing session is complete) and not to the newly modified version of the file (called `sched` after the editing session is complete).

To have your link always pointing to the latest version of `sched`, create a soft link using `ln -s` as shown above.

For more information about links, see the chapter "Managing Links" in *Using Your BSD Environment*.

The UNIX Display-Oriented Editor (vi)

You may prefer to use **vi**, the UNIX display-oriented (visual) text editor, when editing files. When you use **vi**, the position of the cursor on the screen indicates the position within the file. The editor uses commands that let you manipulate text in a variety of ways.

Consult the *UNIX Text Processing* manual for complete details on the use of **vi**. For a much briefer description of usage see the *BSD Command Reference*. To view an online display of the command reference page for **vi**, type the following:

```
% man vi
```

The UNIX Line Editor (ed)

The standard UNIX line editor, **ed**, allows you to edit files on a line-by-line basis. Unlike **vi**, it does not permit you to view large sections of text at once.

Both the *UNIX Text Processing* manual and the *BSD Command Reference* provide more information. To view an online display of the command reference page for **ed**, type the following:

```
% man ed
```

Summary

In this chapter, you learned how to copy and delete files. You also learned how to use directories and links, and about the additional editors available in the BSD environment. The next chapter contains specific information about the C shell.



Chapter 11

Using the C Shell

This chapter explains how to use the C shell, the standard BSD shell. In this chapter, you'll learn about wildcards, symbols that redirect input and output, pipes and filters, and shell scripts.

Command Format

The simplest UNIX shell command consists of a command name by itself, as in the following example:

```
% ls
```

The `ls` (list directory) command lists the contents of your working directory.

Using Command Arguments

Most shell commands accept one or more **command arguments** to specify the object (file, directory, or link) upon which the command acts, for example:

```
% ls your_directory
```

In this example, `your_directory` is the argument. The `ls` command lists the contents of `your_directory`. You must use one or more spaces to separate a command from its arguments, and to separate arguments from each other.

Using Command Options

In addition to command arguments, most shell commands accept **command options** to modify the action of commands. They appear before command arguments. Most options are preceded by a hyphen, although this is not always the case. Check the *BSD Command Reference*, or the online manual page (using the `man` command), for specific usage instructions in the case of each option. If an option is preceded by a hyphen, do not include any space between the hyphen and the option:

```
% ls -l
```

The `ls` command, used with the `-l` option displays the contents of the working directory in “long” format, supplying mode, number of links, owner, size in bytes, and time of last modification for each file in the directory.

Entering Multiple Commands on a Line

To enter more than one command on a single line, separate the commands with a semicolon (;), as in this example:

```
% ls; date
```

This command lists the contents of the working directory, then displays the date. Here’s another example you may want to try:

```
% cd; ls
```

The `cd` (change directory) command changes the working directory to your home directory, and then the `ls` command lists its contents.

Command Line Processing

Most commands are files. When you press <RETURN> to enter one of these commands, the shell searches for a file with the same name as the command you specified. If the shell finds such a file, it loads and executes it.

Next, the shell passes any arguments and options on your command line to the program. After executing the program, the shell displays a new prompt to show that it is ready for your next command. For example, when you enter the command

```
% ls -l my_directory
```

the shell searches for a file named `ls`. The shell then loads the program, passes the option and the argument to the program, and executes it. After the program displays the contents of `my_directory` in a single column, the shell redisplay a prompt on its input pad. You may then enter another shell command.

Command Search Paths

When you enter a command that is the name of a file, the UNIX shell takes a particular route (referred to as a **command search path**) in determining which directories to search for the command file. Display the current search path of the UNIX shell you're using by typing the `echo` command, with `$PATH` as its argument:

```
% echo $PATH
```

NOTE: `PATH` is an instance of an **environment variable**. For more information about environment variables, see *Using Your BSD Environment*.

The following directory names are displayed in response to the `echo` command line above:

```
:/bin:/usr/bin:/usr/ucb:/usr/apollo/bin
```

Let's take a closer look at this information. The sequence of directory names above is stored in the shell variable called **PATH**. The shell looks for command files in this order:

1. Your working directory (**.**), implied by the empty field before the first colon
2. The primary system command directory (**/bin**)
3. A secondary system command directory (**/usr/bin**)
4. A directory containing BSD-specific commands (**/usr/ucb**)
5. A directory containing Apollo-specific commands (**/usr/apollo/bin**)

As soon as the shell finds a filename that matches the command you specify, it attempts to execute the program.

To include other directories in your search path, use the **setenv** command to change **\$PATH**. For example, to add the **~/bin** directory to the end of your default search path, type the following:

```
% setenv PATH :/bin:/usr/bin:/usr/ucb:/usr/apollo/bin:~/bin
```

Using Wildcards

Wildcards are special characters that you may use to represent one or more pathnames. For example, the following command line contains the ***** wildcard:

```
% ls *.bak
```

The expression ***.bak** matches every file that ends in **.bak** (backup versions of text files). Therefore, this command line lists all the files that end in **.bak** in your working directory.

The next example lists all the objects in your home directory beginning with one of the letters a through z.

```
% ls ~/[a-z]*
```

The [...] wildcard, where the ellipsis (...) represents specified characters, matches any of the enclosed characters. A pair of characters separated by a dash (-) matches any character lexically between the pair. Thus [2-5] matches 2, 3, 4, and 5, and [A-Z] matches any capital letter.

Most shell commands that accept pathnames as arguments also accept wildcards. Wildcards are relatively easy to use, but there are more of them than are described here. Do not use wildcards until you are sure that you understand fully how they operate within the particular type of UNIX shell you are using. “Using the C Shell” in *Using Your BSD Environment* gives the most complete information about wildcards.

Redirecting Input and Output

In the previous examples in this book, the shell process read the commands you entered (input) from its input pad, and it wrote the responses to your commands (output) on its transcript pad. The next two sections explain a few examples of special characters that redirect input and output. For complete information about these characters, see the “Using the C Shell” chapter in *Using Your BSD Environment*.

Writing Output to a File

Some shell commands write output to the transcript pad. For example, the following `cat` (catenate) command line, with a filename as an argument, directs its output to the transcript pad:

```
% cat myfile
```

Here, `cat` catenates `myfile` and displays the output on the transcript pad.

To direct the output to a file instead of the transcript pad, use the greater-than symbol (>) and another filename. For example

```
% cat myfile > myfile.out
```

writes the catenated text to **myfile.out**. The shell creates the output file (**myfile.out**) if it doesn't already exist. Note that if **myfile.out** already exists, the shell deletes its old contents first.

Reading Input from a File

To make a shell command read input from a file instead of from the input pad, use the less-than symbol (<) and a filename.

In our next example, we use the **wc** (word count) command, which counts the number of characters (including blank spaces), words (character strings delimited by spaces, tabs, and new lines), and lines contained in the standard input. In this section, you'll learn how to instruct **wc** to read input from a file.

Before executing the **wc** command, create a text file to use as input to the command. To do so, follow these steps:

1. Create an edit file named **sample_file**.
2. When the DM creates the edit window and pad on your screen, type the following lines of text:

```
wc counts the number of words,  
characters, and lines contained in  
the standard input (by default) or in  
a file that you specify.
```

3. Press <EXIT> to close the edit window and pad.

Instruct **wc** to get its input from the file **sample_file** with

```
% wc < sample_file
```

This tells **wc** to supply a count of the lines, words, and characters contained in **sample_file**. Then **wc** writes its output to the transcript pad:

The output represents lines (4), words (23), and characters (129) found in `sample_file`.

Using Pipes and Filters

A **pipe**, represented by the vertical-bar character (`|`), connects the standard output of one shell command to the standard input of another. For example, suppose you want to know how many words two of your files contain, when combined. You could discover this with the two command lines

```
% cat file1 file2 > sample_file
% wc -w < sample_file
```

Clearly, this method is cumbersome. Also, you probably don't want to keep a copy of the union of the two files; all you really need is a report on the total number of words contained in the two files combined. A pipe lets you do both tasks in one simple operation called a **pipeline**:

```
% cat file1 file2 | wc -w
```

In this case, the `cat` command creates an output file only on the standard output and not in your working directory. The `wc` command uses this temporary data as standard input to produce the results on your display.

The length of a pipeline is not restricted to a particular number of operations. However, you should remember that pipes are interpreted by the shell as being unidirectional: that is, commands to the right of the vertical-bar character cannot send data to the commands to the left of it.

Although a regular pipe that has several programs feeding into one final program won't work, you can build a type of "T-joint" in the pipe by using the `tee` command in your pipe. This command tells the shell to transcribe the standard input to the standard output and make copies in the named files.

The `tee` command is particularly useful for saving data being transmitted through the pipe, data that is written over in the process. Suppose you want to check a file called `test` for spelling errors, create a file named `test.errors` that contains the list of misspelled words found in `test`, and create a separate file named `test.num` that provides only a count of the total number of misspellings in `test`. You could use the following command line:

```
% spell test | tee test.errors | wc -w > test.num
```

Since `tee` copies its input to the specified file(s) and to its output, the intermediate output from the execution of `spell` is saved in `test.errors`. A copy of this data passes through the `wc` program, and the result of the execution of `wc` is written into a new file named `test.num`.

A **filter** reads its input, performs some specified operation on it, and then prints the result as output. One useful filter, `fgrep`, searches its input for lines that contain a string specified by you. For instance,

```
% ls | fgrep bak
```

prints all lines from the output of `ls` that contain the string “bak”.

Creating Shell Scripts

A **shell script** is a file that contains a list of shell commands. If you find yourself repeating a sequence of commands over and over again, create a shell script containing the commands. Then you can execute the entire command sequence with a single command.

To create a script, use `<EDIT>` to create the file; type the shell commands, one per line, then press `<EXIT>` to close the file. To execute the script, type the script's filename in the process input window next to the UNIX shell prompt. Be sure to read the remainder of this section for important details about shell script execution.

Let's try an example. Suppose you want to change your current directory to your home directory, and then get a long listing of its contents (using the `-l` option with `ls`). You could accomplish these tasks by entering these shell commands, one after another:

```
% cd
% ls -l
```

The `cd` command sets the working directory to your home directory. The `ls` command displays the name of each object in your home directory. The `-l` option displays each object's permissions, owner, size in bytes, etc.

If you create a shell script containing the `cd` and `ls -l` command lines, you may execute both commands by simply entering the filename of the script. To create a script called `dir`, follow these steps:

1. Go to your home directory by entering the command

```
% cd
```

2. Create a personal directory in which your shell scripts can exist. Typically, your built-in scripts should reside in a `bin` subdirectory of your home directory. Add your personal `bin` to your search path so that the shell looks there for commands as well. Go to your `bin` subdirectory by typing

```
% cd bin
```

NOTE: Now, if you type `pwd` at the UNIX shell prompt, you should see your home directory followed by the name `bin`.

3. Create an edit file named `dir`.
4. When the DM creates the edit window and pad, type the shell commands, one per line, like this:

```
cd my_dir
ls -l
```

5. Press `<EXIT>` to close the edit window and pad.

Because files created with the DM editor are by default not executable, you must also enter the command

```
% chmod +x dir
```

NOTE: The command `chmod` (change mode) is used to modify an object's permissions. For more information, see the chapter "Controlling Access to Files and Directories" in *Using Your BSD Environment*.

Using the script that you created, you can now execute the `cd` and `ls` command lines by typing

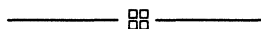
```
% dir
```

Remember that once you have included your personal `bin` directory in your search path, you may execute your shell scripts from anywhere in the naming tree. When you're ready to create more sophisticated scripts, read "Using the C Shell" in *Using Your BSD Environment*.

Summary

In this chapter, you learned more about UNIX shells. You read about how a shell processes your commands and how it locates the command files you request. We also introduced you to some of the more powerful features of UNIX shells, namely wildcards, symbols that redirect input and output, pipes and filters, and shell scripts.

Now that you're familiar with the UNIX shells, read *Using Your BSD Environment* to learn how to perform specific tasks using either the C shell or the Bourne shell. See the *BSD Command Reference* for a brief explanation of how each shell works, or use `man` to view information on the C shell (`csh`), the Bourne shell (`sh`), or the Korn shell (`ksh`).



Part IV

The Aegis Environment

Chapter 12

Introducing Aegis

The Aegis environment is an interface to Domain/OS. All shell examples in this and the following chapters use the Aegis shell.

What Is a Shell?

The shell program provides access to the traditional computing operations, such as printing documents, compiling and running programs, and monitoring system activities. The shell “listens” for commands that you type at your keyboard. Shell commands invoke *utilities*, which are programs that perform the tasks you request.

Using Aegis Commands

When you press <SHELL>, the DM creates a new process running the Aegis shell program, and displays the windows associated with the new shell process. You may create a new shell process at any time and use it to execute programs while your initial shell process is busy performing other tasks.

Move the cursor next to the shell prompt (\$) and enter the **date** command:

```
$ date
```

In the process output window, the shell displays the command you type and then invokes the utility program that displays the date and time.

Getting Help

Domain/OS provides a **help** facility that gives information about Aegis and DM commands. To display introductory information about the help facility, type

```
$ help
```

In a separate window, you'll see an overview of the **help** facility. To remove the window from your screen when you have finished reading the information, press <EXIT>.

Summary

After reading this chapter, you should know how to enter Aegis commands and get help. In the next chapter, you'll learn about commands that manipulate files and directories.



Chapter 13

Working with Files and Directories in Aegis

This chapter explains how to work with directories and how to copy and delete files.

Working with Directories

This section explains some of the most common ways you'll work with directories.

Listing the Contents of Your Working Directory

Use the `ld` (list directory) command to list the contents of a directory. When entered by itself, the `ld` command displays the contents of your working directory.

Listing the Contents of Your Node Entry Directory

You can use a single slash (/) as a shorthand way of referring to your node's entry directory. For example, the following command lists the contents of your node entry directory:

```
$ ld /
```

When you begin a pathname with a single slash (/), the system begins its search in the node entry directory of the node that is physically connected to your display unit. (If your node is diskless, the system begins its search in the entry directory of your node's disked partner.)

You could also list the contents of your node entry directory by typing the `ld` command and a pathname that includes the root directory (//) and your node entry directory name. For example, if you were working on `node_1` in the sample naming tree shown in Figure 13-1, the command

```
$ ld //node_1
```

lists `backup` and `smith`.

To refer to the entry directory on another node, use a pathname that starts with the root directory (//) and includes the name of the other node's entry directory.

Identifying Your Working Directory

To display the name of the directory you're using (your working directory), enter the `wd` (working directory) shell command:

```
$ wd
```

The working directory influences where a process creates or searches for object names (the names of files, directories, or links).

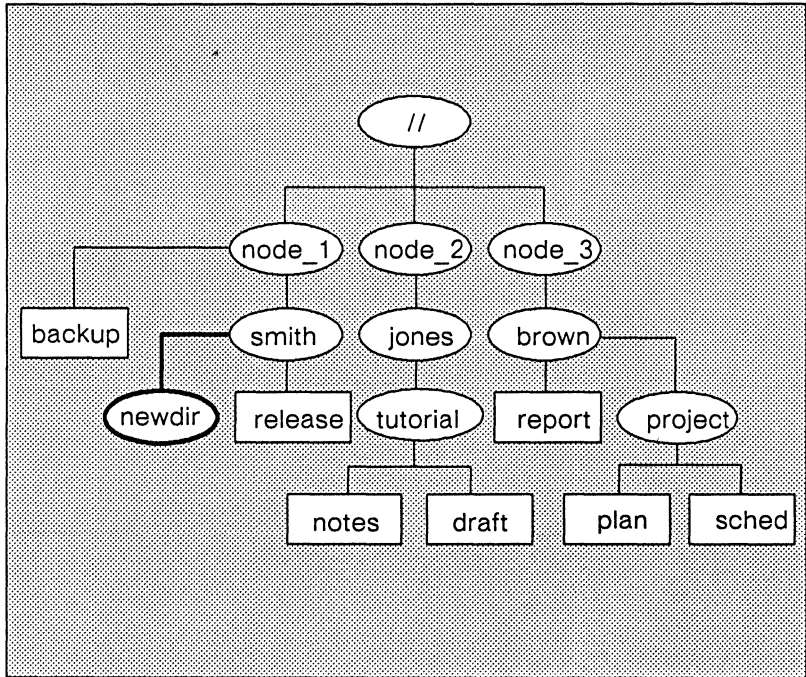


Figure 13-1. A New Directory in the Naming Tree

Changing Your Working Directory

To change the working directory to another directory, use the `wd` command. Type

```
$ wd directory
```

where *directory* is the name of the new working directory.

Creating a New Directory

Use the `crd` (create directory) command to create new directories in your working directory.

For example, suppose your working directory is `//node_1/smith` in our sample naming tree. To create a directory named `newdir` in `//node_1/smith`, type

```
$ crd newdir
```

The newly created directory, `//node_1/smith/newdir`, becomes a subdirectory of the working directory, `//node_1/smith`. (If you enter the `ld` command, `newdir` appears in the list of the working directory contents.) Figure 13-1 illustrates the location of `newdir` in the naming tree.

Listing the Contents of Your Parent Directory

To display the name and contents of the parent directory (the directory one level above the working directory), type

```
$ ld ..
```

You can use several “`..`” symbols to back up more than one level above the working directory. For example, suppose your working directory is `//node_1/smith/newdir`, and you want to change the working directory to `//node_1`. This is accomplished with the line

```
$ wd ../../
```

Displaying Your Naming Directory

To display your naming directory, enter the shell command `nd` (naming directory).

```
$ nd
```

The system uses your log-in directory as the initial naming directory for each process.

Changing Your Naming Directory

To change the naming directory, type the `nd` command followed by the pathname of the new naming directory. For example,

```
$ nd //node_3/brown/project
```

makes `//node_3/brown/project` the new naming directory. You can then use the shorthand symbol “`~`” in place of `//node_3/brown/project`. Thus, the pathname `~/schedule` refers to the file `//node_3/brown/project/schedule`.

Pathname Symbols

You can refer to any object in your network by using a pathname that begins with the root directory (`//`). The system also provides shorthand symbols that you can use in place of longer pathnames. Table 13-1 summarizes pathname starting point symbols.

Table 13-1. Pathname Starting Point Symbols

If your pathname starts with this symbol:	The system begins the name search in this directory:
<code>//</code>	Network root directory
<code>/</code>	Node entry directory
No symbol or <code>.</code>	Working directory
<code>..</code>	Parent directory
<code>~</code>	Naming directory

Figure 13-2 illustrates how you can use different symbols to refer to the same directory in the naming tree.

Remember that the examples in Figure 13-2 assume that the node entry directory is `//node_x` and that the working directory is `//node_x/john`. Each of the pathnames in Figure 13-2 searches for a destination object named `mary`.

The object's full pathname is `//node_x/mary`. The pathname `/mary` instructs the system to look for `mary` in the node's entry directory (`/`). The pathname `../mary` instructs the system to move up to `//node_x` and then look for an object named `mary`.

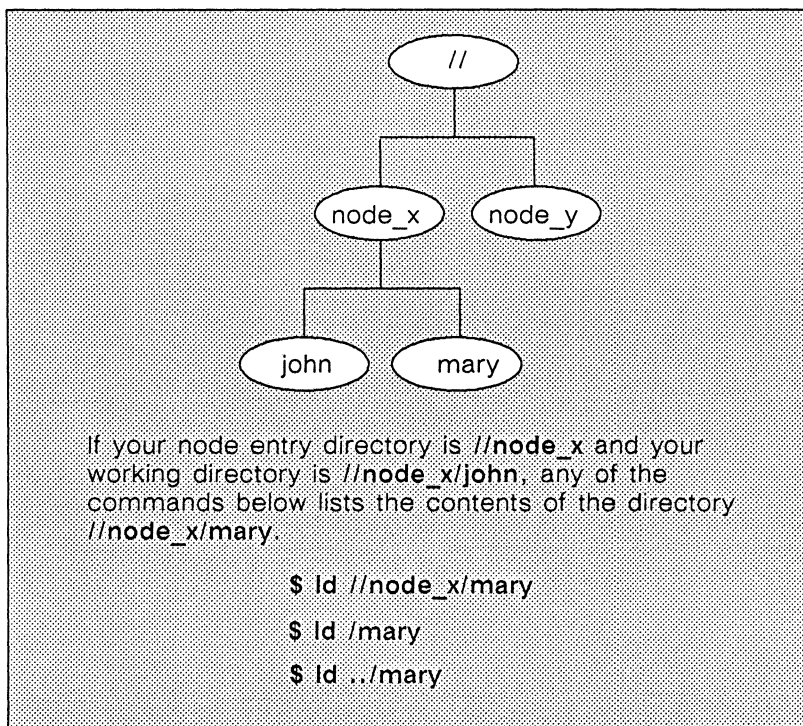


Figure 13-2. Pathnames Starting with `//`, `/`, and `../`

Working with Files

This section explains how to copy and delete files.

Copying a File

The shell command `cpf` (copy file) copies the contents of a file and stores it in another file. You specify the file to be copied and the new file. For example, to make a copy of `sample_edit` and store it in a new file called `copied_file` in your working directory, enter the following command:

```
$ cpf sample_edit copied_file
```

The DM copies `sample_edit` to `copied_file` in the working directory. If you enter the shell command `ld`, you'll see `copied_file` listed in the contents of your working directory.

To copy a file not in your working directory, or to create a copy elsewhere, supply a pathname.

Deleting a File

The shell command `dlf` (delete file) deletes files from directories. To delete a file, enter the `dlf` command and a filename. For example:

```
$ dlf copied_file
```

deletes the file `copied_file` from your working directory. To delete a file anywhere else in the network, supply a pathname.

Using Links

As you use the system, you may find that you frequently access objects with very long pathnames. Instead of typing the long pathname each time you want to use an object, you can create a special object type called a link.

A link is a shorthand name for a pathname. It contains the pathname of another object. When you use a link name as a pathname or as part of a pathname, the system substitutes the name inside the link (the resolution name) for the link name.

For example, let's say you work on `node_1` in our sample naming tree (Figure 13-1), and you often use the file `sched`. To access `sched`, you could type the name `//node_3/brown/project/sched` each time you want to use the file.

However, it is much easier to create a link containing this pathname. Then you can use the shorter link name whenever you want to use `sched`. To create a link, use the shell command `cr1` (create link). To create a link called `mylink` in the working directory, type

```
$ cr1 mylink //node_3/brown/project/sched
```

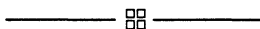
Now you can refer to `//node_3/brown/project/sched` with the name `mylink`. The shell substitutes the resolution name (`//node_3/brown/project/sched`) for the link name (`mylink`).

If you keep links in the naming directory, you can use them regardless of your working directory. To use a link in your naming directory, simply precede the link name with the naming directory symbol (`~`) followed by a forward slash (`/`).

For more information on links, see the chapter "Managing Links" in *Using Your Aegis Environment*.

Summary

In this chapter, you learned how to copy and delete files, and how to create links. Chapter 14 contains more information about Aegis shell commands.



Chapter 14

Using the Aegis Shell

This chapter explains additional information about the Aegis shell and its commands. You'll learn about the parts of a shell command line, how the shell locates and interprets the commands you enter, how to use symbols that have special meaning to the shell, and how to create your own shell commands, called **shell scripts**.

Command Format

The simplest shell command consists of a command name like

```
$ ld
```

This **ld** (list directory) command lists the contents of your working directory.

Using Command Arguments

Most shell commands accept one or more **command arguments** specifying the object upon which the command acts, for example:

```
$ ld your_directory
```

In this example, `your_directory` is the argument. The `ld` command lists the contents of `your_directory`. You must use one or more spaces to separate a command from its arguments, and to separate arguments from each other.

Using Command Options

In addition to arguments, most shell commands accept command options. **Command options** modify the action of commands. They can generally appear either before or after command arguments. Precede each option with a hyphen. Do not include a space between the hyphen and the option, for instance:

```
$ ld -c
```

The `ld` command displays the contents of the working directory; the `-c` option instructs `ld` to list the contents in a single column.

You can use the following options with any shell command:

- `-help` Displays detailed information about the command and its format.
- `-usage` Displays a brief description of the command and its format.
- `-version` Displays the software version number.

Enter the `ld` command with one of the options in the preceding list. The shell displays the information on its transcript pad.

For information about options that you can use with a specific command, check the system help files (type `help` and the command name), or refer to the *Aegis Command Reference*.

Entering Multiple Commands on a Line

To enter more than one command on a single line, separate the commands with a semicolon (;). For example:

```
$ ld; date
```

This command lists the contents of the working directory, then displays the date. Here's another example that you'll probably use often:

```
$ wd ~ ; ld
```

The `wd` command sets the working directory to your home directory, and the `ld` command lists its contents.

Entering Long Command Lines

A `<RETURN>` character usually delimits a command line. To enter a command line that is longer than the current input window, type `@` before pressing `<RETURN>`. This causes the shell to ignore the `<RETURN>` character, and continue the command on the next line of the input pad. For example,

```
$ wd //node/owner@  
$_ /directory
```

sets the working directory to `//node/owner/directory`. The `@` symbol makes the shell ignore the `<RETURN>` character it precedes, and places an underscore (`_`) character in the first position on line two. The underscore indicates that the command begins on the previous line; the shell skips over this character when it reads your command.

Command Line Processing

Most commands are the names of files. When you press <RETURN> to enter one of these commands, the shell searches for a file with the same name as the command you specified. If the shell finds such a file, it loads and executes it as a program.

Next, the shell passes any arguments and options on your command line to this program. After executing the program, the shell displays a new "\$" prompt to show it is ready for your next command. For example, when you enter the command

```
$ ld my_directory -c
```

the shell searches for a file named `ld`. The shell then loads the program it finds in the file `ld`, passes the argument `my_directory` and the option `-c` to the program, and then executes it. After the program displays the contents of `my_directory` in a single column, the shell redisplay the "\$" prompt on its input pad. You can then enter another shell command.

Command Search Rules

When you enter a command that is the name of a file, the shell follows **command search rules** to determine which directories it searches to locate the command file. You can display the current search rules by using the shell command `csr` (command search rules). For example:

```
$ csr
```

A set of command search rules comes with your system software. If you're using these default search rules, the shell displays these directory names when you enter the `csr` command:

```
. ~/com /com /usr/apollo/bin
```

This list of directory names means the shell looks for command files in this order:

1. Your working directory (`.`)
2. Your personal command directory (`~/com`)
3. The system command directory (`/com`)
4. A directory containing Apollo-specific commands (`/usr/apollo/bin`)

You can also use the `csr` command to change the current search rules. (See the chapter “Using the Shell” in *Using Your Aegis Environment*.)

As soon as the shell finds a filename that matches the command you specify, it attempts to execute the program.

The `~/com` directory is a subdirectory that you can create in your naming directory (`~`). When the shell does not find the command file you specify in the working directory, it checks the `~/com` directory. Therefore, `~/com` is a good place to store shell scripts (see the “Creating Shell Scripts” section later in this chapter) and other frequently used programs. You are not required to create a `~/com` directory—no error occurs if it does not exist.

The `/com` directory contains the command files supplied with your system. The shell checks `/com` if it does not find the command you specify in your working directory or in your personal command directory (`~/com`). For example, when you type

```
$ prf myfile
```

the shell looks for the `prf` command in your working directory first, then in your `~/com` directory. Finally, it finds the `prf` command file in `/com` and prints `myfile` on the line printer. (This example assumes that you have not created an object named `prf` in your working directory or in `~/com`, and that you have not changed your search rules.)

Using Wildcards

Wildcards are special characters that you can use to represent one or more pathnames. For example, the following command line contains the `?*` wildcard:

```
$ ls *.bak
```

The `?*` wildcard matches every file that ends in `.bak` (backup versions of text files). Therefore, this command line lists all the files that end in `.bak` in your working directory.

The next example lists all the objects (files, directories, and links) on `node7`.

```
$ ls //node7/...
```

The `"..."` wildcard matches zero or more characters below the starting point in the naming tree. In this example the starting point is `//node7`.

Most shell commands that accept pathnames as arguments also accept wildcards. Wildcards are easy to use, but there are a lot of them. Do not use wildcards until you are sure you fully understand how they work. See the chapter "Using the Shell" in *Using Your Aegis Environment* for complete information about wildcards.

Redirecting Input and Output

In the previous examples, the shell process read the commands you entered (input) from its input pad, and wrote the responses to your commands (output) on its transcript pad. The next two sections explain a few examples of special characters that you can use to redirect input and output. For complete information about these special characters, see the chapter "Using the Shell" in *Using Your Aegis Environment*.

Writing Output to a File

Many shell commands write output to the transcript pad. For example, the following `fmt` (format text) command line, with a filename as an argument, directs its output to the transcript pad:

```
$ fmt myfile
```

Here, `fmt` formats the text in `myfile` and displays the output (the formatted text) on the transcript pad.

To direct the output to a file instead of the transcript pad, use the greater-than symbol (`>`) and another filename. For example:

```
$ fmt myfile > myfile.fmt
```

The `fmt` command formats `myfile` and writes the formatted text to `myfile.fmt`. Now that the formatted text is stored in a file, you can read it or prepare it for printing.

Reading Input from a File

To instruct a shell command to read input from a file instead of the input pad, use the less-than symbol (`<`) and a filename.

Our example uses the `tlc` (transliterate characters) command, which replaces single characters or a range of characters with other characters that you specify. The `tlc` command normally reads input from the transcript pad.

Before entering the `tlc` command, you need to create a text file to use as input to the command. To do so, follow these steps:

1. Create an edit file `replace_chars`.

2. When the DM creates the edit window and pad on your screen, type the following lines of text:

```
TLC will replace the number two with
the number three in the following lines:
2 years ago
2 days from now
2 hours earlier
```

3. Press <EXIT> to close the edit window and pad.

Now instruct `tlc` to get its input from the file `replace_char`. Type the following command line next to the `$` prompt:

```
$ tlc 2 3 <replace_chars
```

The number 2 is the character to be replaced, and the number 3 is the replacement character. The sequence `<replace_chars` instructs `tlc` to perform this replacement task on the text in the file `replace_chars`. The `tlc` command writes its output to the transcript pad, like this:

```
TLC will replace the number two with
the number three in the following lines:
3 years ago
3 days from now
3 hours earlier
```

If you omit the less-than symbol and the input filename, `tlc` waits for you to type the text for the replacement task in the process input window.

To instruct `tlc` to write the output to a file instead of the transcript pad, use the `>` symbol and an output filename. For example:

```
$ tlc 2 3 <replace_chars >replace_chars.new
```

The `tlc` command replaces every occurrence of the number 2 in the file `replace_chars` with the number 3 and writes the output to the file `replace_char.new`.

Creating Shell Scripts

A shell script is a file that contains a list of shell commands. If you find yourself repeating a sequence of commands over and over again, create a shell script containing the commands. You can then execute the entire command sequence with a single command.

To create a script, use <EDIT> to create the file. Type the shell commands, one per line, then press <EXIT> to close the file. To execute the script, you type the script's filename in the process input window next to the \$ prompt.

For example, to change your working directory, then list the contents of the new working directory with special options, enter these shell commands:

```
$ wd my_dir
$ ld -st -dtm
```

The `wd` command sets the working directory to `my_dir`. The `ld` command displays the name of each object in `my_dir`. The `-st` option displays each object's system type (file, directory, or link), and the `-dtm` option displays the date and time you last modified each object.

If you create a shell script containing the `wd` and `ld` command lines, you can execute both commands by simply entering the filename of the script. To create a script called `dir`, follow these steps:

1. Create an edit file named `dir`.
2. When the DM creates the edit window and pad, type the shell commands, one per line, like this:

```
#!/com/sh
wd my_dir
ld -st -dtm
```

The first line tells the system to run the commands which follow in an Aegis shell.

3. Save the file by pressing <SAVE> or <EXIT>.

Because files created with the DM editor are by default not executable, you must also execute the command

```
$ edacl dir -ar userid.%.% -x
```

where *userid* is your user name. Thus, if your user name is “pat”, type

```
$ edacl dir -ar pat.%.%
```

NOTE: The command `edacl` (edit access control list) is used to modify an object’s permissions. For more information, see the chapter “Controlling Access to Files and Directories” in *Using Your Aegis Environment*.

You can now execute both the `wd` and `ld` command lines by typing

```
$ dir
```

Substituting Arguments

The shell script `dir` isn’t very useful because it operates on a single directory (`my_dir`) only. To create a more useful script, we need a way to substitute any directory name for `my_dir`. The shell provides a way to substitute arguments. To do so, edit the `dir` script and change `my_dir` to `^1`. Your results should look like this:

```
wd ^1
ld -st -dtm
```

After you press <EXIT> to close the edit window and pad, instruct `dir` to operate on any directory by supplying the directory name as an argument. For example:

```
$ dir any_directory
```

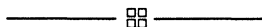
The shell substitutes the first command argument you specify (in this case **any_directory**) for **^1**, and then executes the script. The **dir** command then sets the working directory to **any_directory** and lists the name, type, and date and time of the last modification for each object in **any_directory**.

The previous two sections provided basic information about shell scripts. When you're ready to create more sophisticated scripts, read "Writing Shell Scripts" in *Using Your Aegis Environment*.

Summary

In this chapter, you learned more about the shell and its commands. You read about how the shell processes your commands and how it locates the command files you request. We also introduced you to some of the more powerful features of the shell, namely wildcards, symbols that redirect input and output, and shell scripts.

Now that you're familiar with the shell and the DM, read *Using Your Aegis Environment* to learn how to perform specific tasks. See the *Aegis Command Reference* for detailed command descriptions.



Appendices

Appendix A

Keyboard Command Summary

The following tables summarize the preset key definitions for the different environments. Frequently used DM commands are also included where relevant.

Table A-1. Moving the Cursor

Task	All environments
Move left one character	←
Move right one character	→
Move up one line	↑
Move down one line	↓
Move to start of line	←
Move to end of line	→
Tab right	<TAB>
Tab left	CTRL/<TAB>
Move to DM input window	<CMD>
Move to next window	<NEXT WNDW>

Table A-2. Process Control

Task	Aegis	SysV or BSD
Create a new process and process windows	<SHELL>	<SHELL>
Stop a process, remove input window, and delete pads	CTRL/Z	CTRL/D
Save transcript pad in a file	<CMD>pn <i>pathname</i>	<CMD>pn <i>pathname</i>

Table A-3. Window Control

Task	Aegis	SysV or BSD
Bring window to front or send to back	<POP>	<POP>
Change window size	<GROW>	<GROW>
Move a window	<MOVE>	<MOVE>
Cancel grow or move operation	CTRL/X	CTRL/X
Close pad and save any changes	<EXIT> or CTRL/Y	<EXIT>
Close pad without saving changes	<ABORT> or CTRL/N	<ABORT>
Copy text to input pad	<AGAIN>	<AGAIN>

Table A-4. Creating Edit and Read-Only Pads

Task	All environments
Create edit pad and window	<EDIT>
Create read-only pad and window	<READ>

Table A-5. Moving a Pad under a Window

Task	All environments
Move cursor to first character in pad	CTRL/T
Move cursor to last character in pad	CTRL/B
Move pad by pages	<div style="display: inline-block; border: 1px solid black; padding: 2px; margin-right: 10px;">↑</div> <div style="display: inline-block; border: 1px solid black; padding: 2px;">↓</div>
Move pad by lines	SHIFT/↑ SHIFT/↓
Move pad horizontally by ten characters	<div style="display: inline-block; border: 1px solid black; padding: 2px; margin-right: 10px;">←</div> <div style="display: inline-block; border: 1px solid black; padding: 2px;">→</div>
Move pad horizontally by single characters	SHIFT/↑ SHIFT/↓

Table A-6. Editing a Pad

Task	Aegis	SysV or BSD
Set read/write mode	SHIFT/<AGAIN> or CTRL/M	SHIFT/<AGAIN>
Set insert/overwrite mode	<INS>	<INS>
Delete character at cursor	<CHAR DEL>	<CHAR DEL>
Delete character before cursor	<BACKSPACE>	<BACKSPACE>
Delete next word	<F6>	<F6>
Delete to end of line	<F7>	<F7>
Delete entire line	<LINE DEL>	<LINE DEL>
Copy marked range to paste buffer	<COPY>	<COPY>
Cut marked range and write to paste buffer	<CUT>	<CUT>
Paste text from paste buffer	<PASTE>	<PASTE>
Cancel range selection	CTRL/X	CTRL/X
Undo last command	<UNDO>	<UNDO>
Save pad to file without closing pad	<SAVE>	<SAVE>

Table A-7. Searching and Replacing Text

Task	Aegis	SysV or BSD
Search forward for <i>string</i>	<CMD>/ <i>string</i> /	<CMD>/ <i>string</i> /
Repeat last forward search	CTRL/R	CTRL/N
Search backward for <i>string</i>	<CMD>\ <i>string</i> \	<CMD>\ <i>string</i> \
Repeat last backward search	CTRL/U	CTRL/P
Replace all instances of <i>string1</i> with <i>string2</i> in each line of marked range	<CMD>s/ <i>string1</i> / <i>string2</i> /	
Replace first instance of <i>string1</i> with <i>string2</i> in each line of marked range	<CMD>so/ <i>string1</i> / <i>string2</i> /	

Glossary

Access rights

These rights list the people who can use each object in the network, and specify how each person can use the object (e.g., permission to read, write, and execute the object). The phrase is used in connection with the Aegis environment; UNIX users refer to **permissions**, which are comparable (but not equivalent).

Aegis

The environment developed at Apollo Computer. Also refers to the shell program used to interact with the Aegis environment.

Alarm window

The Display Manager alarm window appears near the bottom of your screen. It displays a small pair of bells when a process displays a message in an output window hidden by an overlapping window.

BSD

The environment based on 4.3 BSD UNIX from the University of California at Berkeley. (*See also* SYSTYPE.)

Command

An instruction that you give to run a program.

Command argument

Information you provide on a command line to describe the object (usually a file or directory) to be operated on by the command.

Command option

Information you provide on a command line to indicate any special action you want the command to take. (*See also* Default.)

Command search rules

See Search path.

Control key sequence

A keystroke combination used as a shorthand way of specifying commands. To enter a control key sequence, hold <CTRL> down while pressing another key.

Current directory (.)

See Working directory.

Cursor

The small blinking box initially displayed in the screen's lower left corner. The cursor marks your current typing position on the screen and indicates which program (shell or DM) receives your commands.

Default

Most programs give you a choice of one or more options. If you don't specify an option, the program automatically assigns one. This automatic option is called the default. (*See also* Command option.)

Directory

A special type of object that contains information about the objects beneath it in the naming tree. Basically, it is a file that stores names and links to files. (*See also* File.)

Disk

A thin, record-shaped plate that stores data on its magnetic surfaces. The system uses heads (similar to heads in tape recorders) to read and write data on concentric disk tracks.

Diskless node

A node that has no disk for storage, and therefore uses the disk of another node. (*See also* Node and Disk.)

Display Manager (DM)

The program that executes commands that start and stop processes, and commands that open, close, move, or modify windows and pads.

DM environment variables

Values set by either the system or the user to determine how the Display Manager handles processes started at login or during command execution.

DM function keys

Single keys that invoke DM commands.

DM input window

The window where you type DM commands (contains the "Command: " prompt).

DM output window

The window that displays output messages from DM commands.

Domain/OS

The Apollo operating system supporting a high-speed communications network connecting two or more nodes. Each node can use the data, programs, and devices of other network nodes. Each node contains main memory, and may have its own disk, or share one with another node.

File

The basic named unit of data stored on disk. (*See also* Directory.)

Filter

A command that reads the standard input, performs a user-specified task, and outputs the result on the standard output.

Hard link

A link that points directly to an object (file).

Home directory (~)

A shorthand way of referring to a frequently-used directory, almost always the **log-in directory**. This is a UNIX term; in Aegis, one refers to the **naming directory**.

Input pad

A **pad** that accepts commands typed at your keyboard.

Input window

The **window** that displays a program's prompt and any commands typed but not yet executed.

Insert mode

This mode lets you change text displayed in windows. Modify text by repositioning the cursor and typing characters.

Link

A special type of object that points from one place in the naming tree to another. (*See also* Hard link and Soft link.)

Logging in

Initially signing on to the system so that you may begin to use it. This creates your first user process.

Log-in directory

The directory in which you are placed when you log in.

Name

A character string associated with a file, directory, or link. A name can include various alphanumeric characters, but never a slash (/) or null character.

Naming directory (~)

The Aegis shell allows you to refer to a frequently-used directory with the tilde (~) symbol. Aegis uses your log-in directory as the initial naming directory. UNIX users generally call this the **home directory**.

Naming tree

A hierarchical tree structure that organizes network objects.

Network

Two or more nodes sharing information.

Network root directory (//)

The top directory in the network. Each node has a copy of the network root directory.

Node

A network computer. Each node in the network can use the data, programs, and devices of other network nodes. Each node contains main memory, and has its own disk, or shares one with another node. (*See also* Diskless node.)

Node entry directory (/)

A subdirectory of the network root directory. The node entry directory is the top directory on each node. Diskless nodes share the node entry directory of their disked partner node. (*See also* Network root directory.)

Object

Any file, directory, or link in the network.

Operating system

The program that supervises the execution of other programs on your node.

Option

See Command option.

Output window

The window that displays a process response to your command.

Pad

A temporary, unnamed file that holds the information displayed in a window. A window can display an entire pad or only part of the pad. (*See also* Window.)

Parent directory (..)

The directory one level above your working directory.

Partner node

A node that shares its disk with a diskless node. (*See also* Diskless node.)

Password

The word you enter next to the "Password:" prompt at log-in time. As you type your password, the system displays periods (.) instead of the letters in your password. You should keep your password secret and change it occasionally in order to protect your account from unauthorized use. (*See also* User account.)

Pathname

A series of names separated by slashes that describe the path of the operating system from some starting point in the network to a destination object. Pathnames begin with the starting point's name, and include every directory name between the starting point and the destination object. A pathname ends with the destination object's name.

Permissions

A set of rights (read, write, execute) associated with an object in the file system. Determines who may use the object. *See also* Access rights.

Pipe

A facility that connects the standard output of a command with the standard input of another command.

Print server

A process that oversees the printing of files submitted to the print queue.

Process

A computing environment in which you may execute programs.

Prompt

A message or symbol displayed by the system to let you know that it is ready for your input.

Regular expression

A string specifier that can help you find occurrences of variables, expressions, or terms in programs and documents. Regular expressions often include wildcards for matching several items.

Root directory

See Network root directory.

Script

A file that you create that contains one or more shell commands. A script lets you execute a sequence of commands by entering a single command (the script filename). (*See also* Shell command.)

Search path

The ordered set of directory names in which a shell looks for the names of commands when they are invoked. Also called search rules.

Shell

A command-line interpreter program used to invoke utility programs.

Shell command

An instruction you give the system to execute a utility program. (*See also* Script.)

Shell metacharacter

Any character that has special meaning to a shell—for example, asterisks, question marks, and ampersands.

Soft link

A link that points to the pathname of an object (file). (*See also* Link.)

Software

Programs, such as the shells and the DM, that allow you to perform various tasks.

Start-up script

A file that sets up the initial operating environment on your node. This file is also known as a “boot script”. (*See also* Script.)

System administrator

The person responsible for system maintenance and security at your site.

SYSTYPE

A DM environment variable that shows your default UNIX environment. Valid SYSTYPES are "sys5" and "bsd4.3". (*See also* DM environment variable.)

SysV

The Domain/OS environment derived from UNIX System V, Release 3, from AT&T Bell Laboratories. (*See also* SYSTYPE.)

Transcript pad

A transcript pad contains a record of your interaction with a process. The process output window provides a view of its transcript pad.

User account

The system administrator defines a user account for every person authorized to use the system. Each user account contains the name the computer uses to identify the person (user ID), and the person's password. User accounts also contain project and organization names, helping the system determine who can use the system, and what resources they can use. (*See also* User ID and Password.)

User ID

The name the computer uses to identify you. Your system administrator assigns you your user ID. Enter your user ID during the log-in procedure when the system displays the log-in prompt. (*See also* User account.)

Utilities

Programs provided with the operating system to perform frequently required tasks, such as printing a file or displaying the contents of a directory. (*See also* Command.)

Wildcards

Special characters that you may use to represent one or more pathnames or other strings of characters. (*See also* Shell metacharacter.)

Window

An opening on the screen for viewing information. Display management software lets you create several windows on the screen. Each window is a separate computing environment in which you may execute programs, edit text, or read text. (*See also* Pads.)

Window legend

The area of a window that displays window status information. For example, the window legend of an edit window contains such information as the pathname of the file you're editing, the letter "I" or "R" indicating edit mode, and the number of the line at the top of the window. (*See also* Insert mode.)

Working directory

The default directory in which a process creates or searches for objects. Sometimes called the current directory.

Index

Symbols are listed at the beginning of the index. Entries in color indicate task-oriented information.

Symbols

- .. (double period), 13-4
- ... (ellipsis), 14-6
- ; (semicolon), 8-2, 11-2, 14-3
- [] (square brackets), 8-5, 11-5
- @ (at sign), 14-3
- * (asterisk), 8-4, 11-4
- / (slash), 10-2
 - in search strings, 5-12
- ^ (caret), 14-10
- | (vertical bar), 8-7, 11-7
- < (less than), 8-6, 11-6, 14-7
- > (greater than), 8-6, 11-5, 14-7
- \ (backslash), in search strings, 5-12
- (tilde), 7-4, 10-4, 13-5
- _ (underscore), 14-3

A

- ABORT key, 5-14
- access rights, 5-4, GL-1
- Aegis environment, GL-1
- AGAIN key, 5-1, 5-4
- alarm window, 2-7
- alarms, 3-9
 - responding to, 3-9
- ap (alarm pop) DM command, 3-9
- arguments, substituting, 14-10 to 14-11
- arrow keys, 2-1, 5-5 to 5-6

B

- BACKSPACE key, 2-4, 5-8
- backup files, 5-14

.bak files, 5-14
Bourne shell, 6-1, 8-1 to 8-11
BSD environment, GL-1

C

cat (catenate) UNIX command,
8-5
cd (change directory) UNIX
command, 10-4
CHAR DEL key, 2-10, 5-8
chmod (change mode) UNIX
command, 8-10, 11-10
CMD key, 2-4
command, GL-1
command arguments, 8-1 to
8-2, 11-1 to 11-2, 14-1 to
14-2, GL-2
command options, 8-2, 11-2,
14-2, GL-2
command search paths, 8-3 to
8-4, 11-3 to 11-4
command search rules, 14-4,
GL-2
CTRL key, 2-9
CTRL/B, 5-4
CTRL/D, 3-9
CTRL/M, 5-4
CTRL/N, 5-12
CTRL/P, 5-12
CTRL/R, 5-12
CTRL/T, 5-4
CTRL/U, 5-12

CTRL/X, 3-5, 3-7, 5-9
CTRL/Z, 3-9
control keys, 2-8 to 2-9, GL-2
controlling, windows, command
summary, A-3
COPY key, 2-8, 5-9
copying
files, 7-8 to 7-9, 10-8 to
10-9, 13-7
text, 5-9
correcting errors, 5-8
cp (copy file) UNIX command,
7-8, 10-8 to 10-9
cp (create process) DM
command, 3-3 to 3-4
cpf (copy file) Aegis command,
13-7
crd (create directory) Aegis
command, 13-3
creating
directories, 13-3 to 13-4
hard links, 10-10 to 10-11
pads, command summary,
A-4
shell process windows, 3-3 to
3-4
shell scripts, 8-8, 11-8, 14-9
to 14-11
soft links, 10-9 to 10-10
crl (create link) Aegis
command, 13-8
csr (command search rules)
Aegis command, 14-4
current directory. *See* working
directory

cursor, 1-3, GL-2
 moving, 2-1 to 2-3
 command summary, A-2
 moving between windows,
 2-7

CUT key, 2-8, 5-10

cutting text, 5-9 to 5-10

cv (create view) DM command,
 5-2

D

Display Manager (DM), 1-4,
 GL-3
 commands, entering, 2-8
 environment variables, GL-3
 function keys, 2-8, GL-3

Domain/Delphi, 1-5

Domain/OS, 1-4

date Aegis command, 6-2

default, GL-2

defining keys, 2-3

deleting

 files, 7-9, 10-9, 13-7 to
 13-9

 text, 5-9 to 5-10

directories, 4-1, GL-2

 creating, 10-5 to 10-8, 13-3
 to 13-4

 listing contents, 13-1

 parent, GL-6

disk, GL-2

diskless node, GL-3

display, landscape, 1-3

dlf (delete file) Aegis command,
 13-7

documentation conventions, vi

E

ed (line editor) UNIX
 command, 7-11, 10-11

edacl (edit access control list)
 Aegis command, 14-10

EDIT key, 5-6

editing, pads, command
 summary, A-5

editors

 DM, 5-1 to 5-18

 ed (UNIX line editor), 7-11,
 10-11

 vi (UNIX screen editor),
 7-11, 10-11

entering multiple commands,
 8-2, 11-2, 14-3

erasing, files, 7-9, 10-9, 13-7

errors, correcting, 5-8

EXIT key, 3-9, 5-4, 5-14

F

fgrep UNIX command, 8-8,
 11-8

files, 4-1

 closing, 5-14 to 5-15

 copying, 7-8 to 7-9, 10-8 to
 10-9, 13-7

 deleting, 7-9, 10-9, 13-7 to
 13-9

 opening for editing, 5-6

 printing, 5-15 to 5-18

files (cont.)

- reading input from, 8-6 to 8-7, 11-6 to 11-10, 14-7 to 14-10
- saving, 5-14
- writing output to, 8-5 to 8-6, 11-5 to 11-6, 14-7

filters, 8-8, 11-8

fmt (format text) Aegis command, 14-7

function keys, DM, 2-8

G

GROW key, 3-4 to 3-5

H

hard links, 10-10, GL-3

help, getting, 6-2, 9-2, 12-2, 14-2

HELP key, 5-17

home directory, 7-4, 10-4, GL-4

I

input

- reading from a file, 8-6 to 8-7, 11-6 to 11-10, 14-7 to 14-10

input pad, GL-4

input window, GL-4

INS key, 2-10, 5-8

insert mode, 2-7, 2-10, GL-4

K

keyboard, 1-3, 2-1 to 2-2

keyboard commands, summary, A-1 to A-6

keys, defining, 2-3

L

ld (list directory) Aegis command, 5-3, 13-1 to 13-3

LINE DEL key, 2-10, 5-8

links, 4-2, 13-8, GL-4

- hard, 10-9
 - creating, 7-10
- soft, 10-9 to 10-10
 - creating, 7-9

ln (link) UNIX command, 7-10, 10-10

log-in directory, 4-3, GL-4

logging in, 2-3 to 2-4

logging out, 2-11

ls (list directory) UNIX command, 5-3, 7-1, 8-1, 10-1 to 10-3

M

man (manual page) UNIX command, 6-2, 9-2

MARK key, 3-5, 5-8

mkapr (make apollo product report) command, v

mkdir (make directory) UNIX
command, 7-5, 10-5

mouse, 2-2 to 2-3
using to change window size,
3-6
using to open read-only
windows, 5-2
using to read file, 5-2 to 5-4

MOVE key, 3-7

moving
the cursor, 2-1 to 2-3, A-2
pads, command summary,
A-4
text, 5-10

N

naming directory, GL-4
changing, 13-5
displaying, 13-4

naming tree, 4-1, GL-5

nd (naming directory) Aegis
command, 13-4

network, GL-5

network root directory, 4-3,
GL-5

NEXT WNDW key, 2-7

node, GL-5

node entry directory, 4-3, GL-5
listing contents, 13-2

O

objects, 4-1

online information, requesting,
3-9

opening files
for editing, 5-6
for reading, 5-2 to 5-4

output
writing to a file, 8-5 to 8-6,
11-5 to 11-6, 14-7

output window, GL-5

overstrike mode, 2-7

P

pads, 3-1, GL-5
edit, 3-2 to 3-3, 5-1
editing, command summary,
A-5
input, 3-2 to 3-3
kinds of, 3-2
moving, command summary,
A-4
read-only, 5-1
transcript, 3-2 to 3-3

parent directory, 4-4, 13-4
listing contents, 10-3

password, 2-4, GL-6

paste buffers, 5-11

PASTE key, 5-10

pasting text, 5-10

pathname, 4-2 to 4-3, GL-6

pathname symbols
Aegis, 13-5
BSD, 10-7
SysV, 7-7

permissions, 8-10, 11-10, GL-6
See also access rights

pipelines, 8-7, 11-7

pipes, 8-7, 11-7, GL-6

pn (pad name) DM command,
3-8

POP key, 3-7

prf (print file) command, 5-15
to 5-18

printing files, 5-15 to 5-18

process, 1-5, GL-6

process control, command
summary, A-3

prompt, 2-3, GL-6

pwd (print working directory)
UNIX command, 7-5, 10-5

R

READ key, 5-2

reading
files, 5-2
input from a file, 8-6, 11-6,
14-7

redirecting input and output, 8-5
to 8-7, 11-5 to 11-7, 14-6
to 14-8

related manuals, v

replacing text, 5-13, A-6

RETURN key, 2-4

rm (remove) UNIX command,
7-9, 10-9

root directory, GL-7
network, 4-3

S

s (substitute) DM command,
5-13

SAVE key, 5-14

saving
files
and closing windows,
5-14
without closing windows,
5-14
transcript pad contents, 3-8

sc (set case) DM command,
5-12

script, GL-7

scripts, 8-8 to 8-10, 11-8, 14-9

scrolling pads, 5-5 to 5-6

search paths, 8-3 to 8-4, 11-3
to 11-4, 14-4 to 14-5

searching for text, 5-12, A-6

selecting text, 5-8 to 5-9

SHELL key, 3-3

shells, 1-4, 2-5, GL-7
creating, 3-3 to 3-4
scripts, 8-8 to 8-10, 11-8 to
11-10

shell variables, 8-3

soft link, GL-7

stopping shell processes, 3-9

substituting
arguments, 14-10 to 14-11
text, 5-13

system administrator, 1-3, 2-4,
GL-8

SYSTYPE, GL-8

SysV environment, GL-8

T

tee command, 8-8, 11-7

text

- copying, 5-9
- cutting, 5-9
- moving, 5-10
- pasting, 5-10
- replacing, command
 - summary, A-6
- searching, command
 - summary, A-6
- searching for, 5-12
- selecting, 5-8
- substituting, 5-13

tilde (-), as home directory symbol, 7-4, 10-4

tlc (transliterate characters)
Aegis command, 14-7

transcript pads, GL-8

- saving contents, 3-8

U

UNDO key, 5-13

utilities, 6-1, 9-1, 12-1

V

vi (visual editor) UNIX
command, 7-11, 10-11

W

wc (word count) UNIX
command, 8-6, 11-6

wd (working directory) Aegis
command, 13-2

wildcards, 8-4 to 8-5, 11-4 to
11-5, 14-6

window legend, 2-7

windows, 1-1, 3-1 to 3-10

- alarm, GL-1
- changing size, 3-4 to 3-6
- control keys, 3-5
- controlling, command
 - summary, A-3

DM, 2-7

moving, 3-7

popping, 3-7

process input, 2-6

process output, 2-6

read-only

closing, 5-4

opening, 5-2 to 5-4

shell process, 3-3

creating, 3-3 to 3-4

working directory, 4-4, GL-2,
GL-9

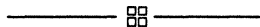
changing, 13-3

identifying, 7-5, 10-5, 13-2

listing contents, 10-3

workstation, 1-1

writing, output to a file, 8-5 to
8-6, 11-5 to 11-6, 14-7



Reader's Response

Please take a few minutes to send us the information we need to revise and improve our manuals from your point of view.

Document Title: *Getting Started with Domain/OS*

Order No.: 002348-A00 Revision: 00

Date of Publication: May 1988

What type of user are you?

- System programmer; language _____
- Applications programmer; language _____
- System maintenance person
- System Administrator Student
- Manager/Professional Novice
- Technical Professional Other

How often do you use the Domain system? _____

What additional information would you like the manual to include? _____

Please list any errors, omissions, or problem areas in the manual by page, section, figure, etc. _____

_____ Your Name

Date

_____ Organization

_____ Street Address

_____ City State

Zip

No postage necessary if mailed in the U.S.



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

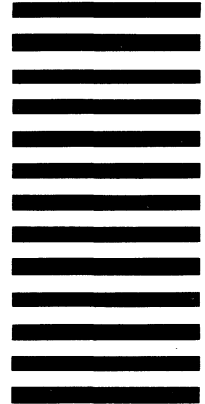
FIRST CLASS

PERMIT NO. 78

CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

APOLLO COMPUTER INC.
Technical Publications
P.O. Box 451
Chelmsford, MA 01824



apollo

Getting Started with Domain/OS (002348-100)



002348-100