

Getting Started With Your DOMAIN System

Order No. 002348
Revision 01
Software Release 7.0

apollo
computer inc.

Copyright © 1983 Apollo Computer Inc.

All rights reserved.

Printed in U.S.A.

First printing: May, 1983
Latest printing: Oct., 1983

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 CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN APOLLO COMPUTER INC. AND ITS CUSTOMERS. NO REPRESENTATION OR OTHER AFFIRMATION OF FACTS 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 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.

Preface

Getting Started With Your DOMAIN System teaches you how to use the keyboard and display, read and edit text, and create and execute programs. You'll also learn how to request system services using interactive commands.

You don't have to be an experienced computer programmer to use this manual. We've geared the material to new users with little or no experience using other computers. We carefully define new terms and try to avoid computer industry jargon. We also include examples you can try at your computer while you read.

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

When you complete this manual, continue with the *DOMAIN System Command Reference Manual*. It contains more advanced information about using the system, and describes each command and feature in detail.

We've organized this manual as follows:

- | | |
|------------|--|
| Chapter 1 | Explains how to use the keyboard and display, and how to issue instructions to the system. |
| Chapter 2 | Describes how the system organizes information. |
| Chapter 3 | Teaches you how to read, create, and edit text. This chapter also explains how to develop and execute programs, and how to print copies of the information stored on the system. |
| Chapter 4 | Documents the Display Manager (DM), a program that executes instructions that pertain to the display or program execution. |
| Chapter 5 | Explains how to use the Shell, a program that executes instructions which request system services. |
| Chapter 6 | Shows you how to protect the information you store on the system from unauthorized use. |
| Appendix A | Describes how to start the computer. |
| Appendix B | Shows how we organize the programs supplied with the system. |
| Appendix C | Lists standard naming conventions. |
| Appendix D | Explains how to read and write to and from a floppy disk. |

*Distributed Operating Multi-Access Interactive Network

Documentation Conventions

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

User Input	Appears in color in <code>this typeface</code> .
System Response	Appears in <code>this typeface</code> .
UPPERCASE	Uppercase characters represent commands.
lowercase	Lowercase characters in command lines represent information that you supply.
[]	Square brackets in command lines enclose items that are optional.
{ }	Braces in command lines enclose a list from which you must choose an item.
	A vertical bar separates items in a list of choices.
< >	Angle brackets enclose the name of a key on the keyboard.
<RET>	The <RET> symbol in command lines indicates when you must press <RETURN>.
CTRL/Z	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 in command lines indicate that you may repeat the preceding item one or more times.
:	Vertical ellipsis points mean we have omitted irrelevant parts of a figure or example.

Problems, Questions, and Suggestions

We appreciate comments from the people who use our system. In order to make it easy for you to communicate with us, we provide the User Change Request (UCR) system for software-related comments, and the Reader's Response form for documentation comments. By using these formal channels you make it easy for us to respond to your comments.

You can get more information about how to submit a UCR by consulting the *DOMAIN System Command Reference Manual*. Refer to the CRUCR (Create User Change Request) command. You can also get more information by typing:

```
$ HELP CRUCR <RETURN>
```

For documentation comments, a Reader's Response form is located at the back of each manual.

Contents

Chapter 1 — Introduction

Logging In	1-1
Changing Your Password	1-4
Understanding the Initial Display	1-4
Moving the Cursor	1-6
Using Directional Keys	1-7
Using the Touchpad	1-7
Using the Mouse	1-8
Entering Shell Commands	1-8
Entering Display Manager (DM) Commands	1-8
Using Display Manager Function Keys	1-9
Using Control Character Sequences	1-9
Correcting Errors	1-9
Looking Inside a Window	1-10
Changing Window Size	1-11
Responding to Alarms	1-11
Stopping a Process	1-12
Ending the Session - Logging Off	1-12
Congratulations	1-12

Chapter 2 — How Does the System Organize Information?

Where Am I?	2-2
The Network Root Directory (//)	2-2
Node Entry Directories (/)	2-2
Your Working Directory	2-2
Your Home Directory	2-3
Parent Directories	2-3
Using Pathnames	2-3
Using the Naming Directory	2-5
Using Links	2-6

Chapter 3 — Using Files

Opening a Window to a File	3-1
Entering Filenames	3-1
Reading a File	3-1
Closing the READ Window	3-3
Editing a File	3-3
Correcting Errors	3-3
Using the Search and Substitute Command	3-5
Using Cut and Paste Commands	3-5
Closing the EDIT Window	3-5
Developing a Program	3-6
Printing a File	3-7

Chapter 4 — Using the Display Manager (DM)

Background Information	4-1
Moving the Cursor	4-2
Using the Arrow Keys	4-2
Using < TAB >	4-2
Using < CMD > and < NEXT WNDW >	4-3
Defining Points and Regions	4-3
Creating a Window: Read and Edit Pads	4-3
Creating an Edit Pad	4-3
Creating a Read-Only Pad	4-4
Managing Windows	4-4
Changing Window Size	4-5
Moving a Window	4-5
Pushing or Popping a Window	4-5
Closing a Window	4-5
Setting Hold Mode	4-6
Copying Text to the Process Input Window	4-6
Moving a Pad Under a Window	4-6
Moving to the Top and Bottom of a Pad	4-6
Moving (Scrolling) Vertically by Pages or Lines	4-7
Moving (Scrolling) Horizontally by Characters	4-7
Editing a Pad	4-7
Defining a Range of Text	4-8
Setting Read/Write, Insert/Overstrike Modes	4-9
Inserting Text	4-9
Deleting Text	4-10
Cutting and Pasting Text	4-11
Searching for Text	4-11
Substituting Text	4-12
Undoing Previous Commands	4-13
Updating an Edit File	4-13
Creating a Process	4-13
Stopping a Process	4-14
Stopping a Shell Process	4-14
Stopping a Program	4-14
Display Manager (DM) Command Summary	4-15

Chapter 5 — Using the Shell

Command Line Processing	5-1
Command Search Rules	5-1
Command Names	5-2
Command Format	5-3
Redirecting Input/Output	5-4
Writing Output to a File	5-4
Reading Data from a File	5-4
Reading Data from Standard Input	5-5
Reading Names from Standard Input	5-5
Using Pipes and Filters	5-6
Using Wildcards	5-6
Writing Shell Scripts	5-7
Substituting Arguments	5-7
Conditional Statements	5-9
In-Line Data	5-9
Displaying Script Commands	5-10
Shell Command Summary	5-10

Chapter 6 — Controlling Access to Files and Directories

Using the Access Control Commands	6-1
Initial Access Control Lists (ACLs)	6-3
Creating Protected Subsystems	6-4

Appendix A — Starting the Node

Appendix B — Initial Directory and File Structure

Appendix C — Standard Suffixes

Appendix D — Using a Floppy Disk

Inserting a Floppy Disk	D-2
Initializing a Floppy Disk	D-4
Mounting a Floppy Disk	D-5
Dismounting and Removing a Floppy Disk	D-5

Illustrations

1-1	The Portrait Display	1-2
1-2	The Landscape Display	1-3
1-3	Window Position on a Portrait Display	1-5
1-4	Window Position on a Landscape Display	1-6
1-5	Cursor Control Keys	1-7
1-6	The DN4xx and DN6xx Keyboard	1-10
1-7	The DN300 Keyboard	1-10
1-8	Window Over a Pad	1-11
2-1	The Naming Tree	2-1
2-2	Sample Pathnames	2-4
2-3	Pathnames Starting with /, //, and \ Symbols	2-5
3-1	Sample Reading Session	3-2
3-2	Sample Editing Session	3-4
3-3	A Simple Program	3-6
6-1	Initial Access Control Lists	6-4
B-1	The Node Entry Directory (/) and Subdirectories	B-1
B-2	The System Software Directory (/SYS)	B-2
B-3	The Display Manager Directory (/SYS/DM)	B-2
B-4	The Network Management Directory (/SYS/NET)	B-3
D-1	A Disk Storage Option Floppy Disk Drive	D-1
D-2	A Cabinet-Mounted Floppy Disk Drive	D-2
D-3	Loading a Floppy Disk in a Cabinet-Mounted Floppy Drive	D-3
D-4	Loading a Floppy Disk in a Disk Option	D-3
D-5	Initializing a Volume	D-4

Tables

2-1	Pathname Starting Point Symbols.....	2-6
5-1	Standard Shell Command Abbreviations.....	5-2
5-2	Special Characters for Interactive Use	5-4
6-1	Access Rights for Files and Directories	6-5
6-2	Abbreviations for Commonly Assigned Rights	6-5
C-1	Standard Name Suffixes.....	C-1

Chapter 1

Introduction

The DOMAIN System is a high speed communications network connecting two or more of our computers, called **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. Ideally, each system user has his or her own node.

The node you're using includes a keyboard and a color or monochromatic 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.

The operating system is a program that supervises the execution of other programs on your node. It includes a number of programs to perform frequently required tasks (such as displaying the date or printing a document). These programs are called **utilities**. To invoke a utility you type a **command** (instruction) at your keyboard. A program called the **Shell** “listens” for these commands. When you type a Shell command, the Shell invokes the appropriate utility program. For example, when you type DATE the Shell invokes the utility program that displays the date.

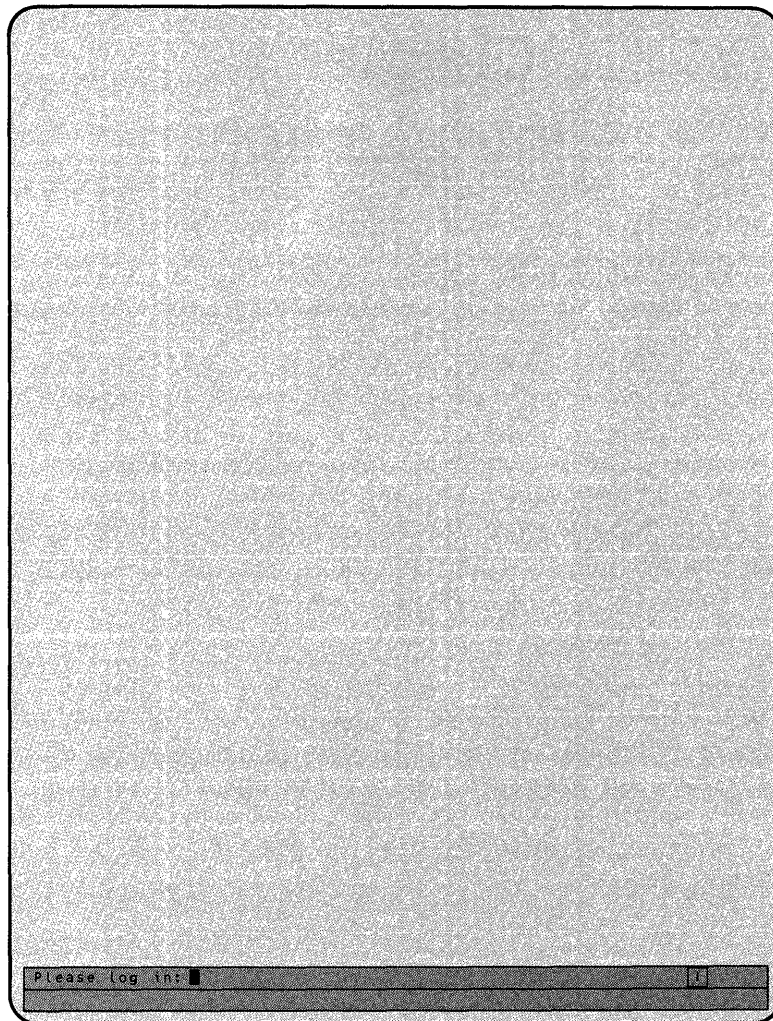
The operating system also includes a program called the **Display Manager (DM)**. The DM “listens” for commands that open, close, move, or modify windows. Certain windows provide a computing environment in which you can execute programs. This computing environment is called a **process**. The DM also “listens” for commands that start and stop processes.

This chapter introduces the DOMAIN system and explains how to issue DM and Shell commands. The best way to master this material is to try each example at your node *while* you read.

Logging In

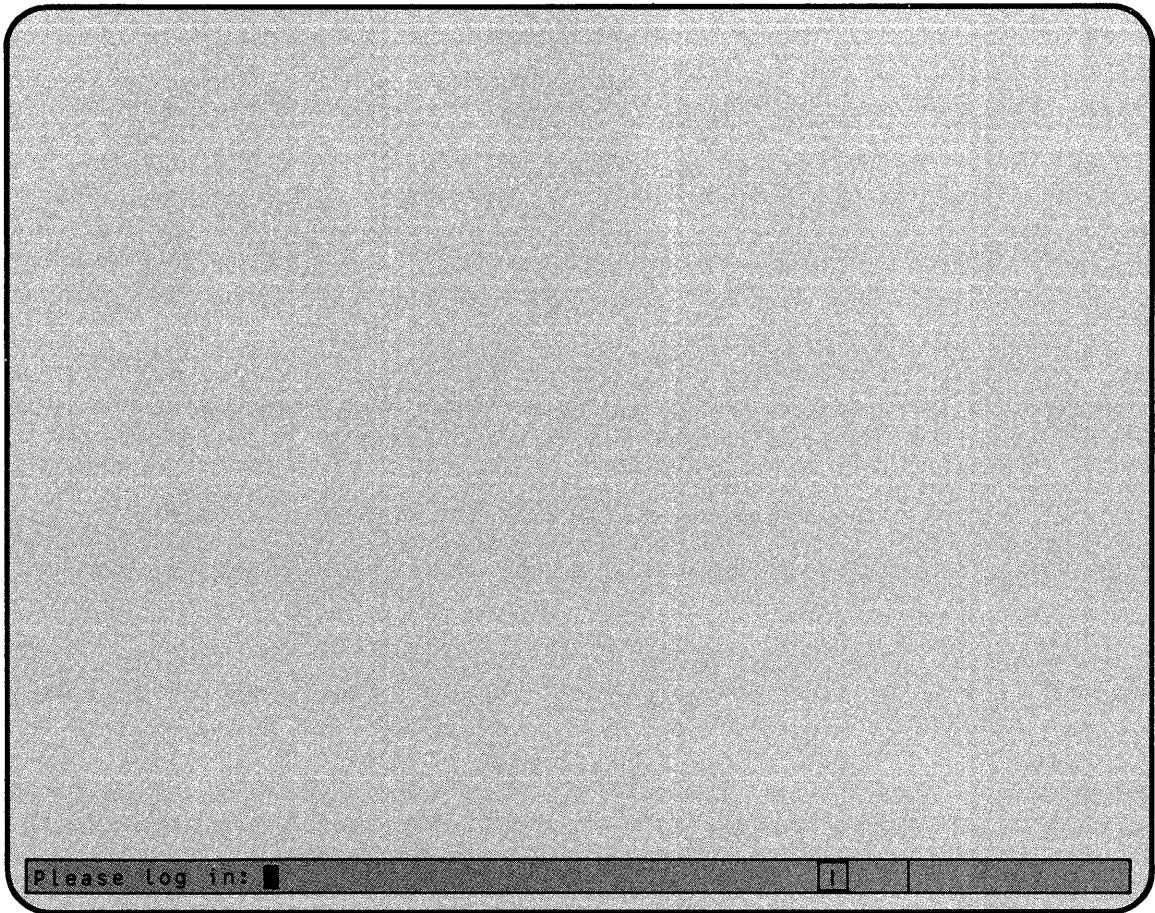
First, turn on your node. (If you're not sure how, see Appendix A.) The display you're using resembles the landscape (horizontal) display in Figure 1-1 or the portrait (vertical) display in Figure 1-2. If your screen looks blank, press any key. The system automatically shuts off the video display if it is idle for more than 15 minutes.

Figure 1-1: The Portrait Display



The message "Please log in: " appears at the bottom of the screen. This message is called a prompt. Programs use prompts to indicate that they are ready for your command. The "Please log in:" prompt indicates that the system is waiting for you to enter your username and password. If you don't know what names to enter, ask your System Administrator (the person responsible for system maintenance and security at your installation). He or she 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 (username), and his or her password. If security is important at your installation, user accounts might also contain project and organization names. The system uses this information to determine who can use the system and what resources they can use.

Figure 1-2: The Landscape Display



To log in, follow steps one through three.

1. Enter the log in (L) command, a blank space, and your username. Press **<RETURN>** (shown in examples as **<RET>**) to submit this information to the system. (Press **<RETURN>** whenever you wish to submit the line you've typed to the system.)

For example:

```
Please log in: L USERNAME <RET>
```

2. If your System Administrator says that project and organization names are required, also type these names before you press **<RETURN>**. For example:

```
Please log in: L USERNAME.PROJECT.ORGANIZATION <RET>
```

3. Now the system requests your password. It does not display the password you enter, but displays a dot for each character in the password. For example:

```
Password: ... <RET>
```

If the system finds a user account that matches the names you supply, it displays a message at the bottom of your screen in the following format:

```
Logged in as user.project.org date time
```

If the system cannot find a user account that matches the names you supply, it repeats the log in prompt and displays this message

```
l name [project [org]] [-p] [-h]
```

If you receive this message, you are not using a valid username and password. Ask your System Administrator for help.

Changing Your Password

You can change your password anytime you log in. After you enter your username(s), enter a space, type -p, and press <RETURN>. For example:

```
Please log in: L USERNAME.PROJECT.ORGANIZATION -p <RET>
```

After you supply your current password, the system displays the following prompt:

```
Enter new password:
```

Supply the new password and press <RETURN>. Next, the system prompts you to repeat your new password (to insure that you enter it correctly). Use this password the next time you log in.

If security is important at your installation, you should avoid using obvious passwords such as your username or your initials. If security is not a high priority, then you can use short or blank passwords. To change your password to a blank, enter a space in quotation marks. For example:

```
Enter new password: " " <RET>
```

To enter a blank password when you log in, just press <RETURN>.

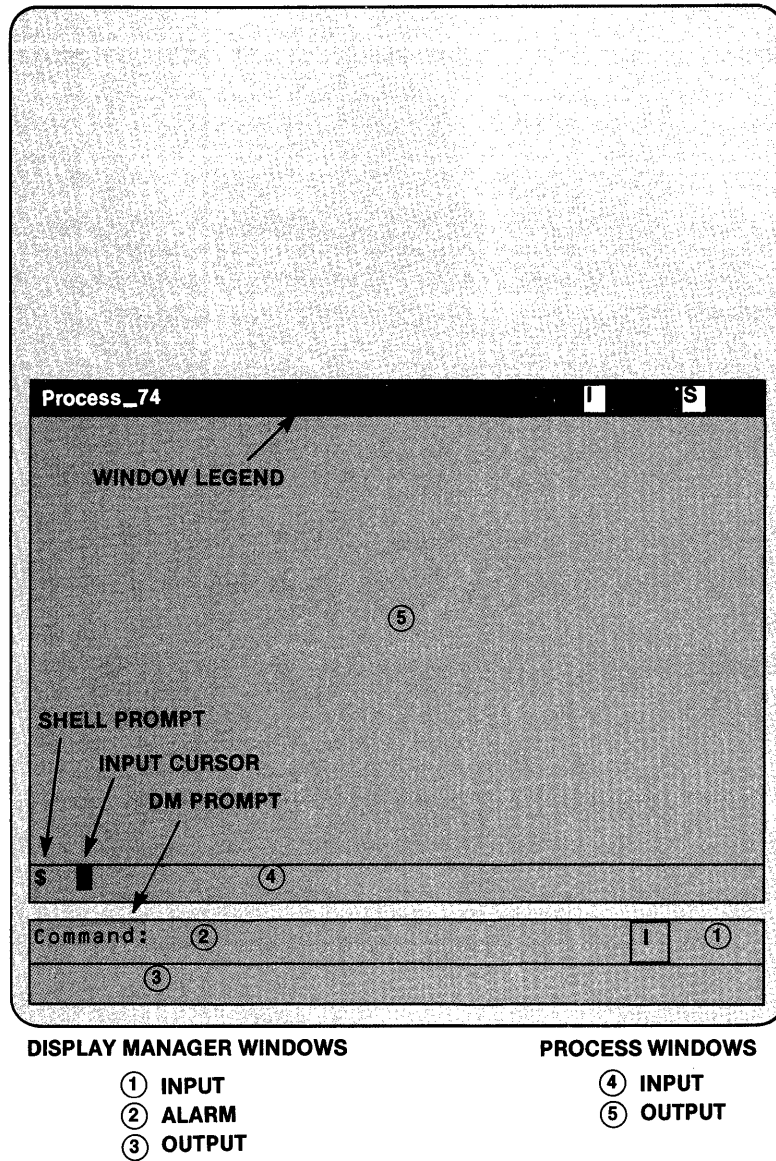
Understanding the Initial Display

After you log in, the system creates a process and presents some windows on your display. The position of these windows depends upon the type of display you're using. Figure 1-3 shows window position on a portrait display. Figure 1-4 illustrates window position on a landscape display.

Notice the small, blinking box in the screen's lower, left corner. It is called the **cursor**. You move the cursor to select the window and position where the system will display the commands you type at your keyboard. The cursor position also determines which program receives your commands. The DM doesn't recognize Shell commands, and the Shell doesn't recognize DM commands. So, before you enter a command, move the cursor into the appropriate window.

To enter Shell commands (commands that invoke other programs) move the cursor into the window that contains the dollar sign \$ prompt, and type the command. To enter DM commands (commands that manipulate windows and processes) move the cursor into the

Figure 1-3: Window Position on a Portrait Display

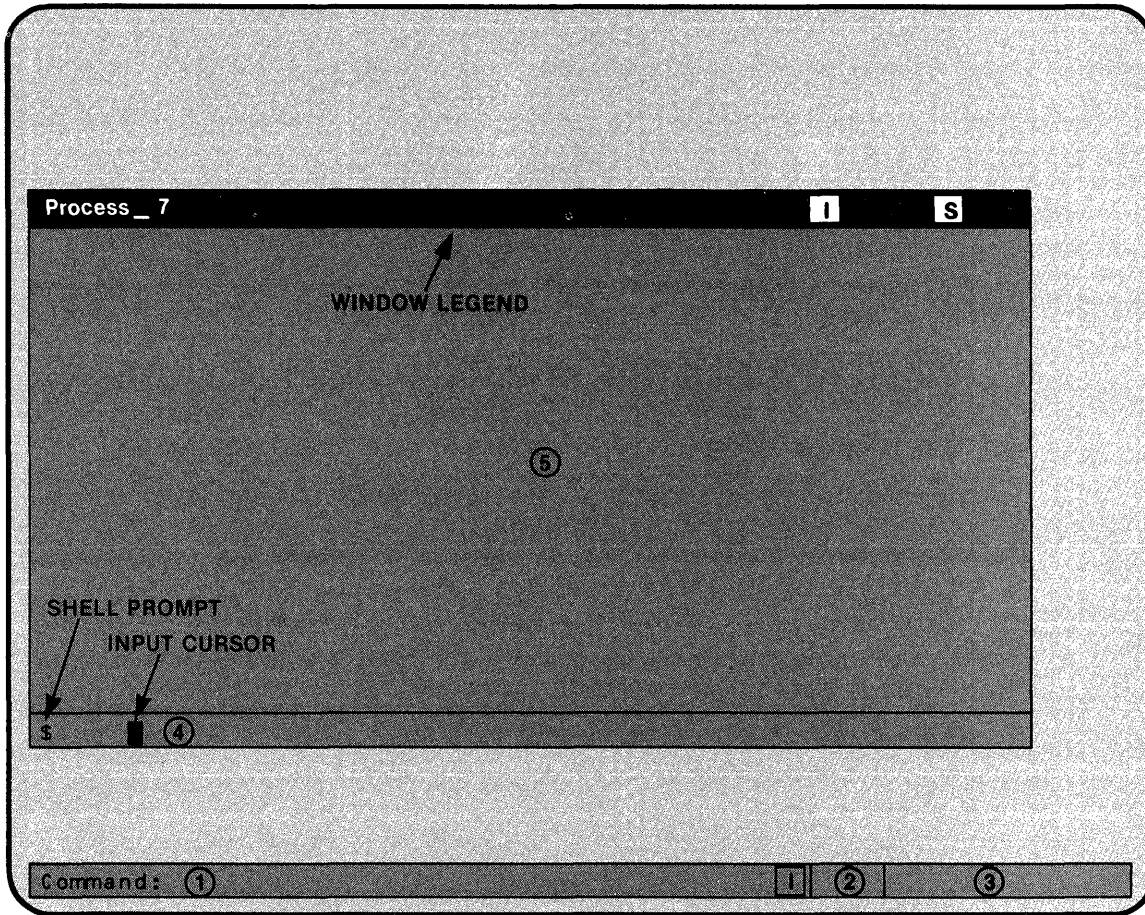


window that contains the “Command:” prompt and type the command. (The following section “Moving the Cursor” explains how to move the cursor from one window to another.)

Windows that contain a program’s command prompt are called **input windows**. Input windows display the commands you type. Programs read commands from input windows. (Refer to Figures 1-3 and 1-4 as you read about windows in the next few paragraphs. The numbers we use to identify windows correspond to these figures.)

The DM displays its “Command:” prompt in the input window at the bottom of your screen (1). The Shell displays its dollar sign (\$) command prompt in the process input window (4) above the DM input window. When you use the Shell to execute another program, that program may display its own command prompt in the process input window. The process input window displays the Shell prompt (\$) when the Shell is “listening” for your next instruction.

Figure 1-4: Window Position on a Landscape Display



- ① INPUT
- ② ALARM
- ③ OUTPUT

- ④ INPUT
- ⑤ OUTPUT

Output windows display the process's response to your command(s). The response can be information you requested, a process status report, or an error message. The DM's output window (3) appears at the bottom of your screen. The process output window (5) appears above the process input window. The process displays your command(s) in its output window before responding to them.

The DM **alarm window** (2) displays a small pair of bells when a process displays a message in an output window that is hidden by an overlapping window. The "Responding to Alarms" section, which appears near the end of this chapter, explains more about this.

Moving the Cursor

To move the cursor use directional keys, a touchpad device or a mouse. The touchpad and mouse are optional pieces of equipment. Directional keys are available on all keyboards.

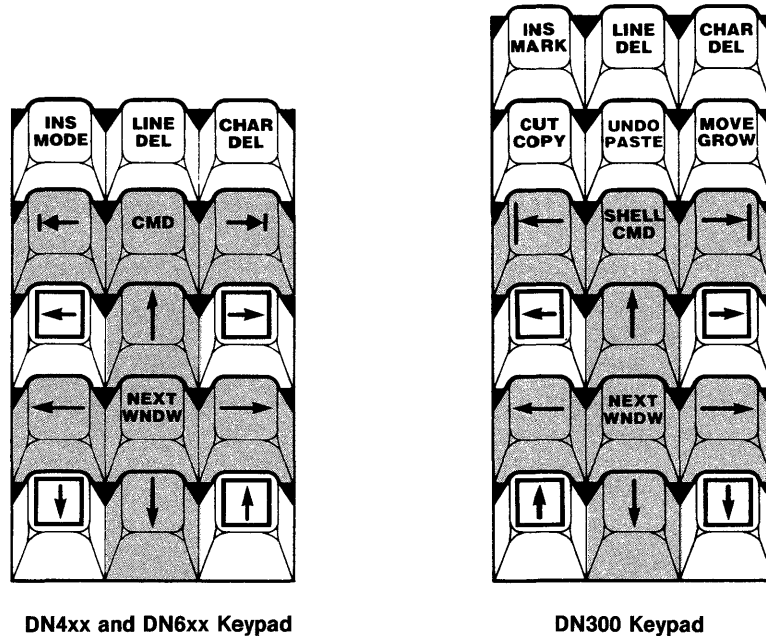
Using Directional Keys

You can use any of the keys highlighted in Figure 1-5 to move the cursor. To move the cursor using 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 a line displayed beneath the cursor, use the keys labeled $\rightarrow|$ and $| \leftarrow$.

To position the cursor next to the DM command prompt, press $\langle \text{CMD} \rangle$. To move the cursor into the process input window, press $\langle \text{NEXT WNDW} \rangle$. You may have to press $\langle \text{NEXT WNDW} \rangle$ more than once to move the cursor into the process input window.

Figure 1-5: Cursor Control Keys



Using the Touchpad

If you have a touchpad, you'll find it on the right side of your keyboard. The touchpad is made of a pressure-sensitive material. When you press the material the touchpad transmits data to the DM, and the DM moves the cursor.

The Shell command `TOUCH PAD MODE (TPM)` lets you control the way the DM interprets information it receives from the touchpad. The *DOMAIN System Command Reference Manual* describes Shell commands in alphabetical order. Refer to the reference manual's TPM command description for further details about the `TOUCH PAD MODE` command.

Take care not to puncture or scratch the touchpad's conductive material. Never use a sharp object on the touchpad. Scratches or punctures can move the cursor into undesired screen positions.

Using the Mouse

If you have a mouse, you can position the cursor by moving the mouse across a flat surface. A small ball bearing in the base of the mouse detects motion, the device transmits data to the DM, and the DM moves the cursor. The TPM command also controls mouse characteristics.

You can use the keys on the mouse to manipulate windows, read files and execute DM commands. Later in this chapter, you'll learn how to expand, shrink, and shuffle windows with the mouse. Chapter 3 shows how to use the mouse to read files. Chapter 4 explains how to redefine mouse keys to execute other DM commands.

Entering Shell Commands

Now that you know how to position the cursor, try entering some DM and Shell commands. The examples in this book include command prompts (\$ and Command:). We include the command prompt to help you determine which input window to enter the command in. Don't type the prompt, just enter the command line shown in color.

We show commands in uppercase letters for emphasis. You can enter commands in upper- or lowercase. Remember to press <RETURN> at the end of each command line. Don't worry if you make a typing mistake when you enter a command. You won't damage the system, you'll simply cause it to display an error message on the screen. If you notice your mistake before you press <RETURN>, you can correct it using <BACKSPACE>. <BACKSPACE> works like the same key on a typewriter, except that it deletes characters as it moves the cursor back towards the beginning of the line. (The "Correcting Errors" section later in this chapter explains other ways to correct mistakes.)

Let's try entering a Shell command. Move the cursor next to the Shell command prompt (\$) and then enter the DATE command. For example,

```
$ DATE <RET>
```

Press <RETURN> to submit the command. The Shell invokes the utility program that displays the date on your screen.

To display a list of commands type HELP.

Entering Display Manager (DM) Commands

To enter DM commands move the cursor into the window that contains the "Command: " prompt and type the command. For example, press <CMD> to move the cursor into the DM input window and type

```
Command: RS <RET>
```

After you press <RETURN> the RS command refreshes the entire screen. You'll see the display blink as the DM clears the screen and redraws windows.

There are two other ways to invoke DM commands from your keyboard. You can press a function key, or enter a control character sequence (CTRL/key).

Using Display Manager Function Keys

You usually won't type DM command names at your keyboard. Instead, you will use certain single keys, called DM function keys that invoke DM commands. We provide files of standard key definitions in the /Sys/DM directory. If the keys on your keyboard don't work as we describe, your start-up files probably point to the wrong set of key definitions. If this happens, ask your System Administrator for help.

Let's try using a function key. Press <SHELL>. It issues the DM command that creates a new process and displays the windows associated with the new process. You can create a new process at any time. You may use it to execute programs while your initial process is busy performing other tasks.

Next, try using the function key <POP>. It executes the DM command WINDOW POP (WP). WP puts the window that contains the cursor on top of the other windows on your screen. If the window that contains the cursor is already on top, the WP command places it underneath the other windows displayed. If you have a mouse, its center key also executes the WINDOW POP (WP) command.

Using Control Character Sequences

Often you'll use control character sequences to invoke DM commands. For example, the control character sequence CTRL/P provides another way to execute WP (window pop). To issue a control character sequence such as CTRL/P, hold <CTRL> down while you type <P>. If the cursor appears in the top window, CTRL/P places that window behind all the other windows.

Correcting Errors

The keys listed next help you correct any errors you might make typing commands (Figures 1-6 and 1-7 highlight these and other keys):

- CHAR DEL
- LINE DEL
- BACKSPACE
- INS (or INS MODE)

<CHAR DEL> deletes the character at the current cursor position, and <LINE DEL> deletes the entire line. <BACKSPACE> works like the same key on a typewriter, except that it deletes characters as it moves the cursor back towards the beginning of the line.

Notice the letter I following the DM input window. The letter also appears in the process's window legend. It indicates the DM and process input windows are operating in **insert mode**.

Insert mode enables you to change command lines in the input window by repositioning the cursor and inserting characters. The rest of the line moves right as you insert additional characters. The system inserts the text you enter rather than overstriking the existing command line. To overstrike the command line, turn off insert mode by pressing <INS>. (On some keyboards this key is labeled INS MODE.)

Figure 1-6: The DN4xx and DN6xx Keyboard

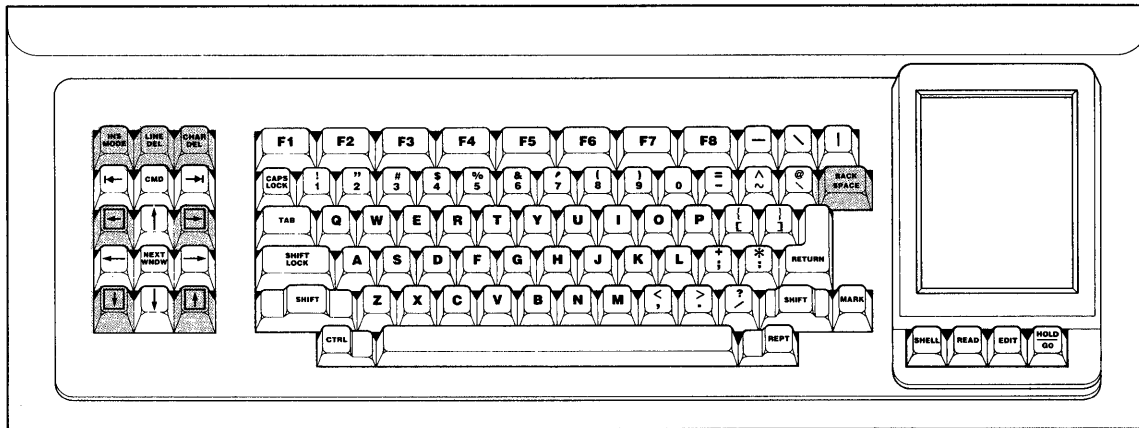
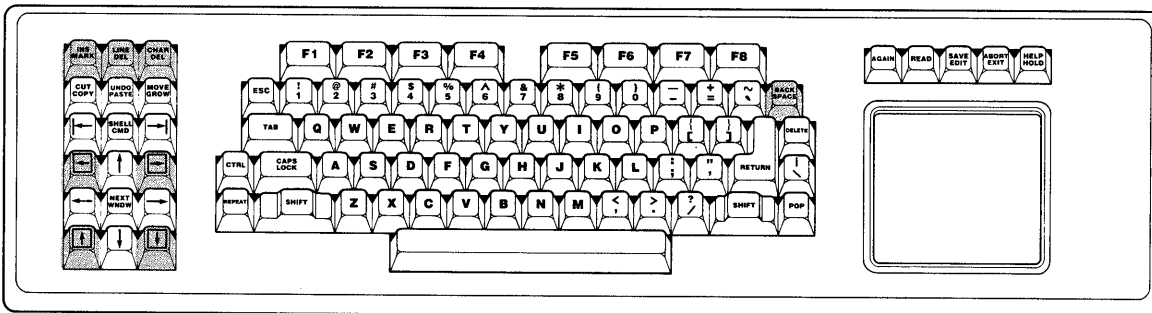


Figure 1-7: The DN300 Keyboard



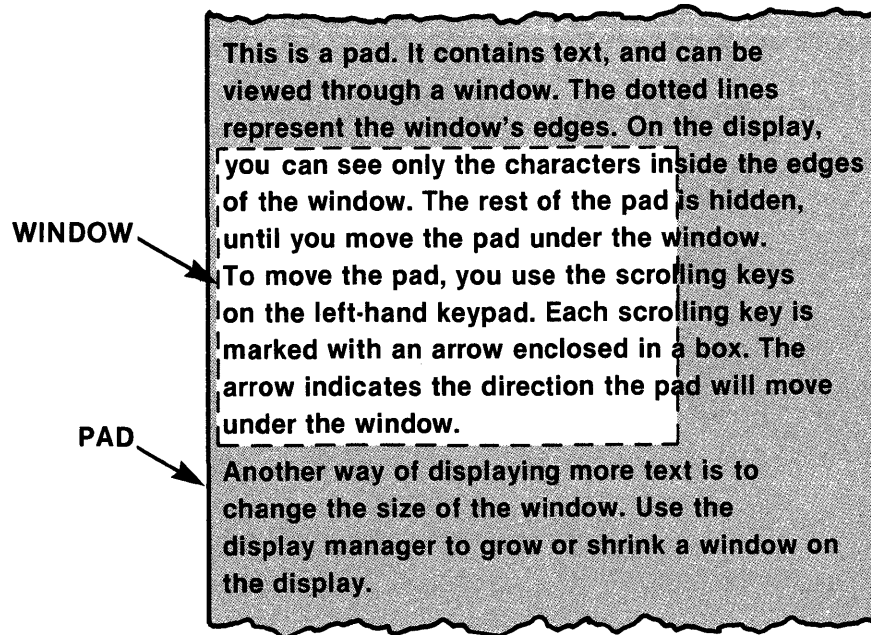
Looking Inside a Window

A window provides a view of a pad. The window can present the entire pad, or only show part of the pad. (See Figure 1-8.) The process output window provides a view to its transcript pad. The transcript pad contains a record of your interaction with the process. It includes every line you input, and every message the process returned.

You can use control character sequences to move the cursor to the top or bottom of a pad. Position the cursor anywhere inside the window, and press CTRL/T. This moves the cursor to the top of the pad. CTRL/B moves it to the bottom.

You can use DM function keys to move the pad beneath the window. To move the pad up, down, right or left use the directional keys highlighted in Figures 1-6 and 1-7.

Figure 1-8: Window Over a Pad



Changing Window Size

Follow these steps to enlarge a window:

1. Place the cursor near the window corner you wish to move.
2. Press `<MARK>`.
3. Move the cursor to the desired new corner.
4. Press `<MOVE/GROW>` or `CTRL/G`.

The window grows to the size you specify by following the above steps.

You can also change window size using a mouse. Place the cursor near the corner you wish to move. Now press the left-most mouse key and hold it down. When you push the key down, the DM executes the `MARK` command. When you release the key, the DM issues the `GROW` command. While you hold the left key down, move the cursor to the desired new corner. Now release the key.

Responding to Alarms

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, and type

```
$ HELP command names <RET>
```

Quickly press `<POP>` again and send the top window to the bottom of the stack.

If this exercise worked, the DM displays two small bells in its alarm window. If your system has a speaker, the DM also emits an alarm tone. The alarm informs you that the system is responding in a window you can't see. Issue the DM command AP (Alarm Pop) to look at the window that requested your attention.

Command: AP <RET>

After you press <RETURN> the DM presents the hidden window.

Stopping a Process

When you finish experimenting, enter CTRL/Z (an end-of-file character) in the process input window. CTRL/Z 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. To remove this window from your screen, keep the cursor within the window and press CTRL/N. To save the transcript pad contents, issue the DM command PN before you press CTRL/N. For example, press <CMD> and type

Command: PN save_file <RET>

The PN command makes the temporary transcript pad a permanent file named save__file. (The filename you specify must be subordinate to your node's entry directory. Chapter 2 explains the network directory structure.) Press CTRL/N when you're ready to delete the window to the transcript pad from your screen.

Ending the Session — Logging Off

When you're ready to end the session, **log off** the system. Logging off prevents others from using your user account. Log off if your node is in a public place.

To log off type the following command in the DM input window:

Command: L0 <RET>

If you've already stopped each process, the DM displays its original "Please log in:" prompt. If you do not stop processes before logging off, the DM stops them for you.

Congratulations

Nice work. You have mastered the system's basic tools. You can log in and off, create and control windows and execute DM and Shell commands. Chapter 2 explains how the system organizes information. Chapter 3 explains how to read, edit, and print information stored on the system. Chapter 3 also shows you how to develop and execute programs. Chapters 4 and 5 provide further information about the DM and Shell. Chapter 6 explains how to protect software from unauthorized use. Continue reading when you are ready to learn more about DOMAIN processing.

Chapter 2

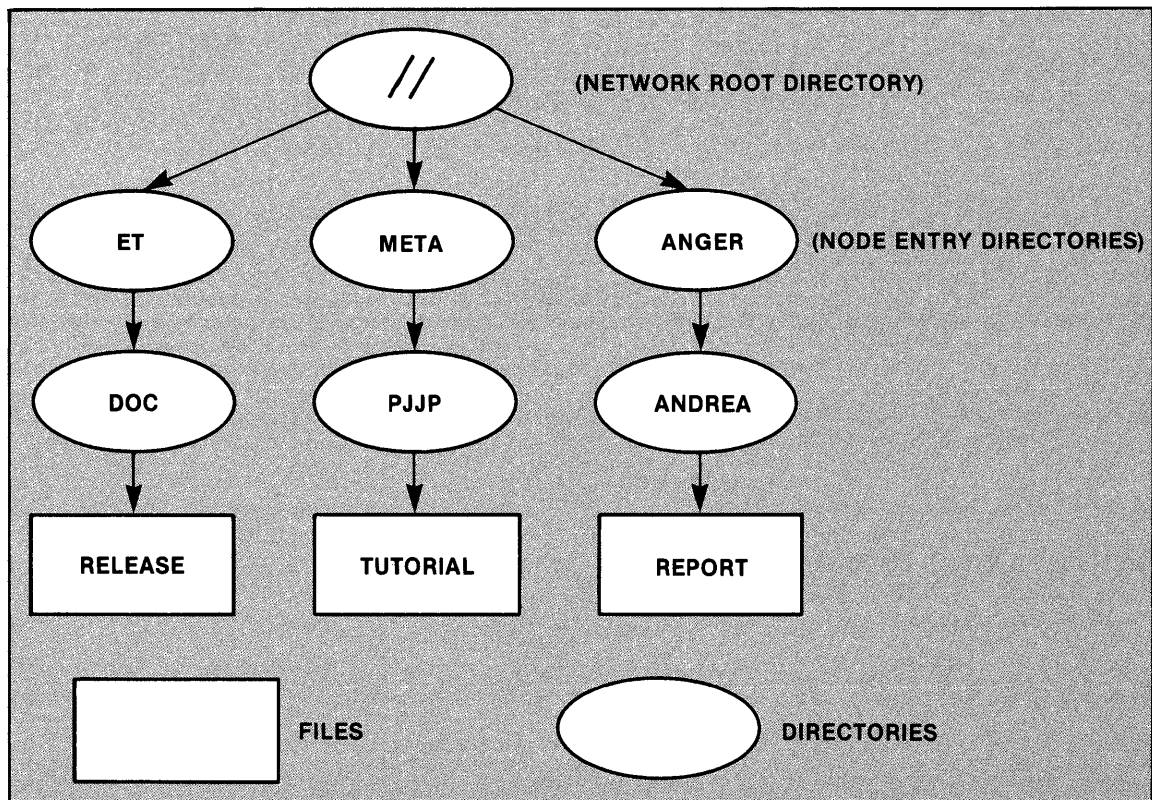
How Does the System Organize Information?

The system organizes 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 a picture — anything you like.

The system organizes related files in **directories**. You can also create directories, and decide which files to store in them. We will explain how to create files and directories later in this chapter. The system keeps track of its files and directories by arranging them 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 2-1 shows a sample naming tree. (Appendix B illustrates how the naming tree organizes system software.)

Figure 2-1: The Naming Tree



The naming tree includes every file and directory in the network. Each directory in the naming tree appears above the files and subdirectories that it contains. Naming tree components are called **objects**. In addition to files and directories, the naming tree includes a third type of object — links. A **link** contains the name of another network object. When the system uses a link, it replaces the link name with the object name the link contains. We will explain how to create and use links later in this chapter. For now, just think of links as a special object type that enables you to take a detour from one part of the naming tree to another.

Where Am I?

Perhaps Figure 2-1 would be more meaningful if it included a “You Are Here” sign. Let’s work our way from the top of the naming tree, the **network root directory**, to the directory you work in when you log in — your **initial working directory**.

The Network Root Directory (//)

The double slashes (//) at the top of the naming tree refer to the network’s top directory, the root directory. The root directory is a list of directory names. It contains the name of each network node’s top directory. To display the root directory’s contents, type

```
$ LD // <RET>
```

You can use the Shell command LD (LIST DIRECTORY) to list the contents of any directory.

Node Entry Directories (/)

The top directory on each node, called the **node entry directory** is a subdirectory of the network root directory. To display your node’s entry directory name type

```
$ LUSR -me <RET>
```

This example includes the “-me” **command option**. Most commands allow you to modify command execution by specifying one or more options. An option follows a command name, and consists of a hyphen and one or more letters. One or more spaces separate a command from an option. No space appears between the hyphen and the option name.

The Shell command LUSR (LIST_USER) with the -me option returns the node’s entry directory name and your username. If you’re using a diskless node, your node uses the entry directory of a disked partner node. (If we omit the command option -me, LUSR lists all network users and their respective node entry directories.)

Your Working Directory

To display the name of the directory you’re using type

```
$ WD <RET>
```

The Shell command WD (WORKING DIRECTORY) returns the name of the current **working directory**. A process’s working directory determines where it creates or searches for

object names (the names of files, directories, or links). For example, if you instruct the process to create a directory named “Newdir” using the Shell command CRD (CREATE DIRECTORY), the Shell process creates the new directory in the current working directory. Therefore, the new directory becomes a subdirectory of the current working directory, and it appears beneath the current working directory in the naming tree.

You can change the working directory to another directory using the WD command. Type

```
$ WD newdir <RET>
```

where newdir is the name of the new working directory.

Your Home Directory

Your initial working directory is called your **home directory**. Your user account contains your home directory name. When you log in, the system sets the working directory to the home directory specified in your user account.

You can change your home directory anytime you log in. After you enter your username(s), enter a space, type -h, and press <RETURN>. For example

```
Please Log in: L USERNAME.PROJECT.ORGANIZATION -h <RET>
```

After you supply your current password, the system prompts you to enter your new home directory’s name. It changes the home directory specification in your user account, and sets your initial working directory to this directory.

Parent Directories

A **parent directory** is the directory above the current working directory. Type

```
$ LD \ <RET>
```

to display the parent directory’s name and contents. The backslash character refers to the directory one level above the current working directory.

Using Pathnames

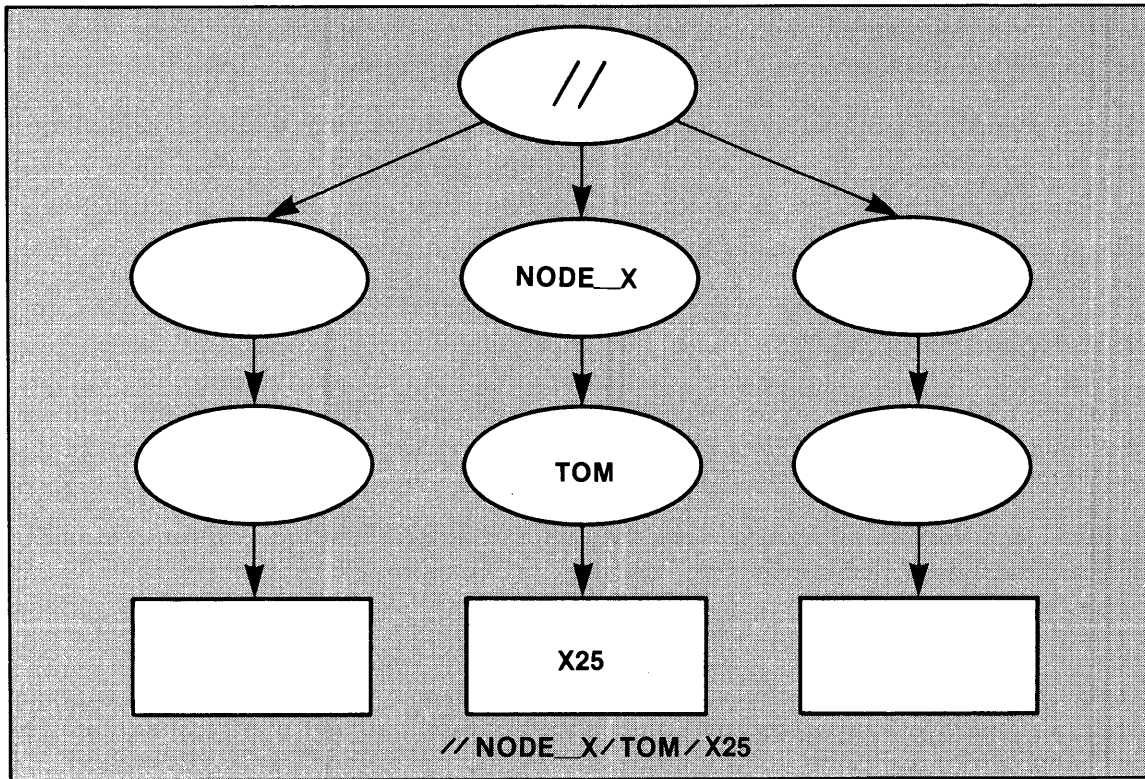
A **pathname** describes the path the operating system must take to get from some starting point in the naming tree to a destination object. A pathname begins with the starting point’s name, and includes every directory name between the starting point and the destination object. A pathname ends with the destination object’s name. Slashes separate names within a pathname. A pathname may not exceed 256 characters, including the slashes.

You can refer to any object in your network using a pathname that begins with the network root (//). For example,

```
//Node_X/Tom/X25
```

describes the path from the top of the naming tree to the file X25. (See Figure 2-2.)

Figure 2-2: Sample Pathname



This pathname instructs the system to:

1. Begin its file search in the list of all nodes (the network root directory)
2. Find the node named Node__X
3. Look in the directory TOM for the file X25

Not all pathnames start with the network root directory. If you only supply a filename (for example, just X25), the system starts its file search in the current working directory.

To refer to your node's entry directory, use a single slash (/). For example,

```
$ LD / <RET>
```

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

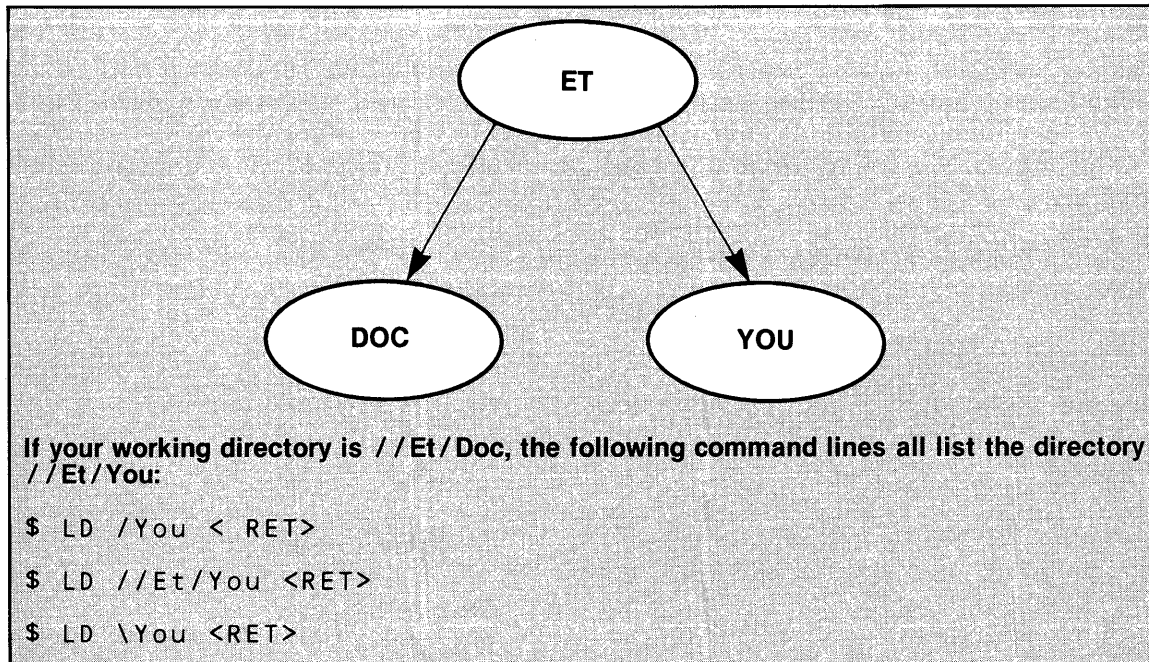
To refer to the entry directory on another node, use a pathname that starts with the network root directory (//) and includes the name of the other node's entry directory. For example,

```
$ WD //Meta <RET>
```

If you begin a pathname with a backslash (\), the system starts its search in the working directory's parent directory. Figure 2-3 illustrates an example.

How Does the System Organize Information?

Figure 2-3: Pathnames Starting with /, //, and \ Symbols



Remember, the example in Figure 2-3 assumes the working directory is //Et/Doc. The pathname /You instructs the system to look for an object named “You” in the node’s entry directory (/). The pathname “\You” instructs the system to move up to //Et and then look for an object named “You.” The object’s full pathname is //Et/You.

Use adjacent backslashes to cause the system to back up more than one directory in the naming tree.

Using the Naming Directory

In addition to its working directory, each process uses a **naming directory**. Like the working directory, the naming directory points to a certain destination directory. You can display or set the destination directory at any time using the Shell command ND (NAMING DIRECTORY). For example, to display the current naming directory type

```
$ ND <RET>
```

The system uses your home directory as the initial naming directory. To change the naming directory, type

```
$ ND dirx <RET>
```

where dirx specifies the new naming directory. After you set the naming directory, you can use the shorthand symbol “~” in place of the naming directory name. For example, if the naming directory is

```
//Nodex/Dirx/Subdirx
```

the pathname ~ Filex refers to

```
//Nodex/Dirx/Subdirx/Filex
```

Chapter 5 explains how to create your own Shell commands. We suggest that you create ~COM, a subdirectory of the naming directory, and keep your personal commands in it. When the system cannot find a command you specify in your working directory, it searches ~COM. This allows you to use your personal commands regardless of your current working directory.

Table 2-1 summarizes pathname starting point symbols.

Table 2-1: 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
~	Naming directory
\	Parent directory
no symbol or .	Working Directory

Using Links

A link is shorthand for a pathname. It is a special object type that contains the name of another object. Links allow you to take a detour from one part of the naming tree to another. 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. To create a link, use the Shell command CRL (CREATE LINK). Type

```
$ CRL mylink /Doc/Ltu <RET>
```

to create a link called mylink in the current working directory. Now you can refer to /Doc/Ltu with the name mylink. The Shell substitutes the resolution name (/Doc/Ltu) for the link name (mylink).

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

Chapter 3

Using Files

This chapter describes how to create, read, edit and print files. It also explains how to develop and execute programs.

Opening a Window to a File

Press <EDIT> or <READ> to instruct the DM process to open a window to a file. Use <READ> to view an existing file, and use <EDIT> to create a new file or to change the contents of an existing one. You cannot modify text displayed in a READ window, but you can change a READ window into an EDIT window by pressing CTRL/M. (You must have the read or write access rights to the file you specify. Chapter 6 documents access rights.)

Entering Filenames

After you press <EDIT> or <READ> the DM prompts you to specify the name of the file you wish to use. When you press <READ> you must specify a filename that already exists. When you press <EDIT> you can specify an existing filename, or create a new file by responding with a filename that does not already exist.

You can construct a new name using 1 to 32 alphanumeric characters, periods (.), underscores (_), or dollar signs (\$). All names must begin with a letter or a dollar sign. (See Appendix C for a list of standard filename suffixes.)

Reading a File

After you press <READ> the cursor moves into the DM input window and the DM displays the message

```
Read file:
```

You must specify a file that already exists. If you specify a filename that does not exist, the DM issues the error message

```
(CV) filename - File not found
```

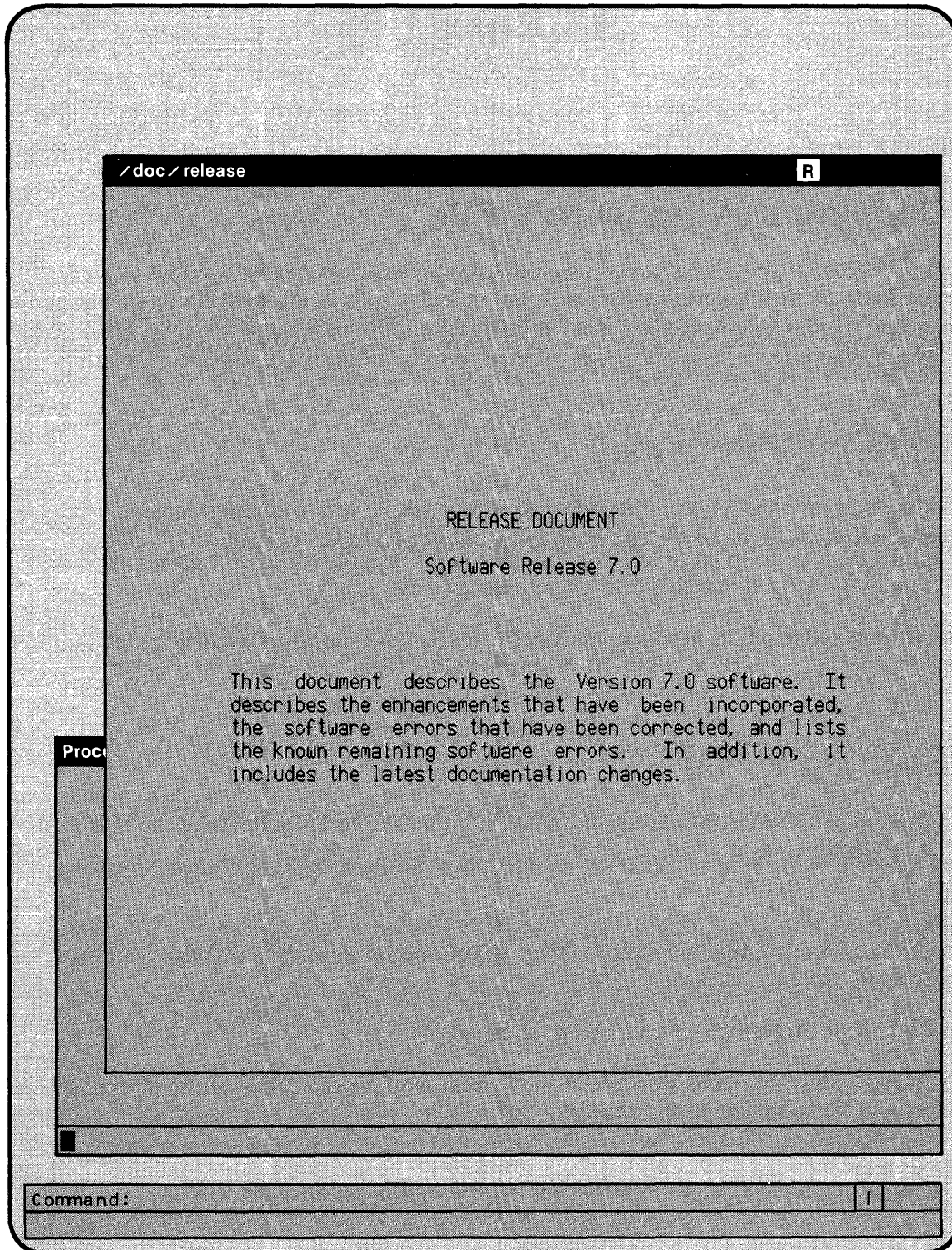
You can supply the name of a file in the current working directory, or use a pathname to refer to a file anywhere in the network.

Each node contains a file called /Doc/Release which describes the current software version. To read the file, respond as follows (you can use upper- or lowercase letters)

```
Read file: /Doc/Release <RET>
```

The DM displays the file /Doc/Release. (See Figure 3-1.)

Figure 3-1: Sample Reading Session



The right-most mouse key also opens files for reading. To use this key, position the cursor next to the name of the file you wish to read. (You may have to use the Shell command LD to display the filename.) Press the right-most key, and the DM creates a read-only window to that file.

Closing the READ Window

When you finish using /Doc/Release, execute the following steps to close the file's window:

1. Move the cursor into the file's window
2. Press CTRL/N

Editing a File

After you press <EDIT> the cursor moves into the DM input window and the DM displays the message

```
Edit file:
```

You can enter any new or existing filename. To create a file called Sample_Edit in your current working directory, type:

```
Edit file: Sample_Edit <RET>
```

When you press <RETURN> the DM opens an EDIT window to the newly-created Sample_Edit file. Notice that the pathname appears in the window legend. Enter the text shown in Figure 3-2. Include the errors highlighted in our example, and feel free to introduce a few new mistakes yourself.

Correcting Errors

You can move the cursor to any text position using arrow keys or the touchpad. Once you've positioned the cursor, use the following keys to correct errors:

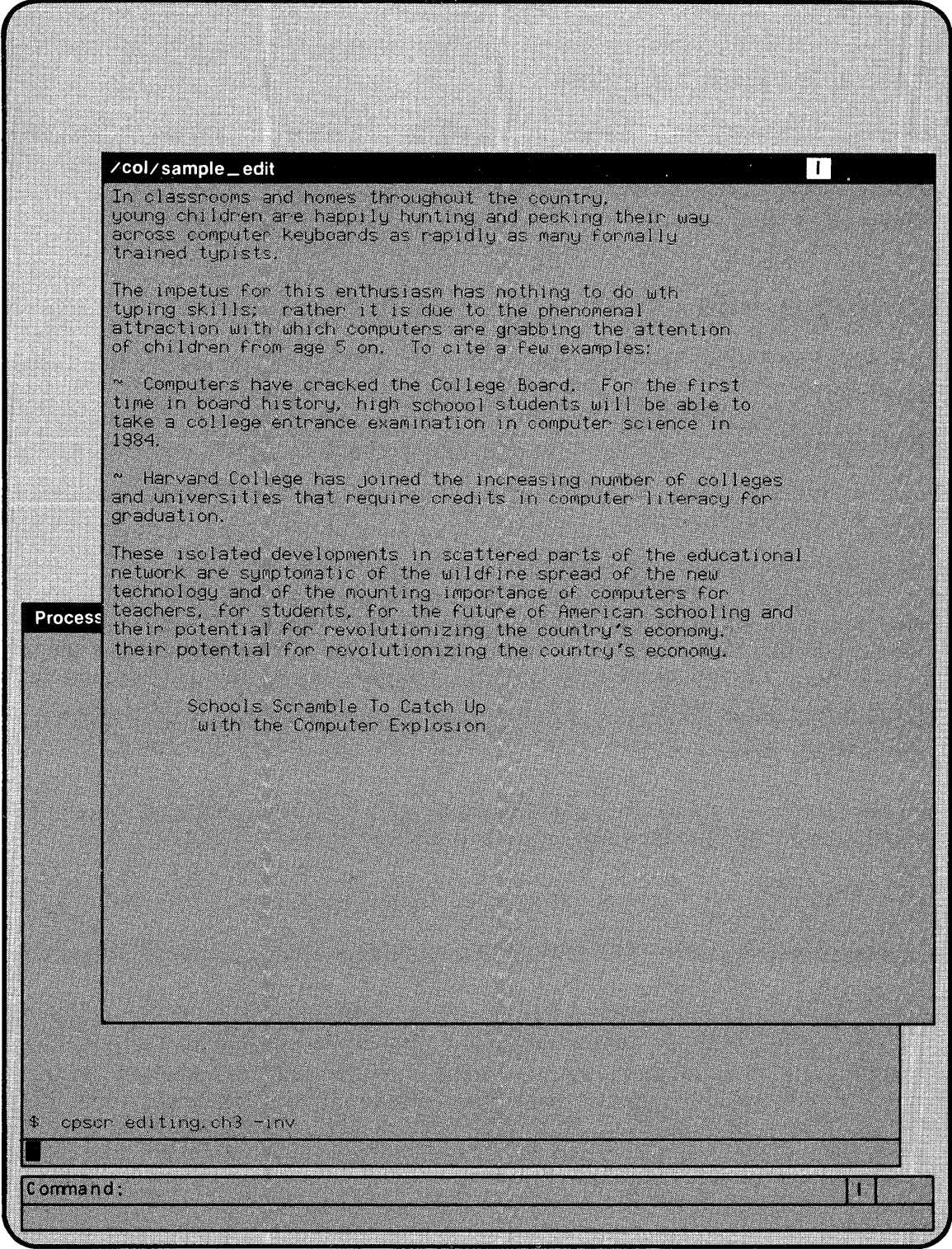
- BACKSPACE
- INS (or INS MODE)
- CHAR DEL
- LINE DEL

Let's start by correcting the errors highlighted in Figure 3-2. We omitted the letter "i" from the word "with." Position the cursor between the letters "w" and "th" and insert the letter. 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. To turn insert mode off and on, press <INS> or <INS MODE>.

Move the cursor to the next typographical error highlighted in Figure 3-2. Use the CHAR DEL key to delete the extra "o" in "school".

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

Figure 3-2: Sample Editing Session



Using the Search and Substitute Command

The DM also provides an easy-to-use search and substitute command. You can use it to search a file or part of a file for a text string, and to replace the string with a new string. First, you must define the range of text you wish to search and substitute. Then, you can use the DM search and substitute command to perform the replacement.

For example, to replace every tilde (~) in Sample__Edit with the letter “o” you must:

1. Define the range of text you wish to search and substitute. The range for this search/substitute operation is from the top to the bottom of the file. To communicate this information to the DM:
 - 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>.Steps A and B tell the DM that the range for the next DM command begins at the top of the file and ends at the bottom.
2. Next, issue the following search and substitute command:

Command: S/~ /o/ <RET>

The DM replaces every tilde (~) in the file with the letter “o.” If you had not marked a range, the DM would have searched only the line that was to the right of the cursor when you pressed <CMD> in step 1-B.

Using Cut and Paste Commands

The DM allows you to delete portions of text from a file, and paste them in a different location in the same file, or in another file. You must define a range of text before you execute the DM cut command.

Let’s try a cut and paste operation. Move “Schools Scramble To Catch Up with the Computer Explosion” from the bottom to the top of the file. To define the range 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 and press <CUT>. (On DN4xx and DN6xx keyboards use CTRL/E.)

<CUT> (or CTRL/E) deletes the text in the range you specified, and copies it into a temporary file called a paste buffer. The DM writes all deleted text into this buffer; however, it saves only the text deleted during the *last* DM operation. Therefore, don’t delete *anything* else until you reinsert the paste buffer contents. Otherwise, you will lose the text you are attempting to move.

To copy the contents of the temporary paste buffer into the top of the file, move the cursor to the top of the file and press <PASTE> (on DN4xx and DN6xx keyboards use CTRL/O).

To copy text into the paste buffer without deleting it from your file, press <COPY> (on DN4xx and DN6xx keyboards use CTRL/C).

Closing the EDIT Window

When you’ve finished editing the file, leave the cursor in the window and press CTRL/Y. This closes the EDIT window and saves Sample__Edit.

To abort an editing session, use CTRL/N instead of CTRL/Y. If you've created a new file during the editing session, CTRL/N ends the session without creating the file. If you've edited an existing file during the session, CTRL/N does not make any of the changes you've entered, and saves the old version of the file. When you press CTRL/N, the DM displays the message

```
File modified. OK to quit?
```

Enter Y or YES to abort the editing session, and N or NO to continue.

Developing a Program

Figure 3-3 shows a simple FORTRAN program which we created using the Display Manager's editing capabilities. To create a program, press <EDIT> and type the program's name next to the "Edit File:" prompt in the DM's input window. Enter your program's statements, and press CTRL/Y to close the edit window.

Figure 3-3: A Simple Program

```
/doc/hello.ftn I 1
C      This is a simple FORTRAN program
C      which prints a simple message.
C      For further details, see Chapter 3 in
C      'Getting Started With Your DOMAIN System'
10     FORMAT ('Welcome to Apollo Computer.',//)
20     FORMAT ('We do windows.',//,
+         'For a demonstration, '//
+         'Press <EDIT> and '//
+         'type any filename next to "Edit file:", '//
+         'and press <RETURN>.')

      PRINT 10
      PRINT 20
      END
```

To keep things simple, we've supplied a sample program for you. Press <EDIT> and type /Doc/Hello.ftn next to the "Edit file:" prompt. After you review the program, press CTRL/Y to close the edit window.

Now you're ready to compile HELLO.FTN. To do this, use the Shell command FTN. Type

```
$ FTN hello.ftn <RET>
```

If you typed everything correctly, the FORTRAN compiler creates a binary file called HELLO.BIN. The compiler also returns a message telling you that there were no errors or warnings in the program unit. If there were errors, you could edit the program again and correct them.

If the program used one or more subprograms, the next step would be to “bind” the main programs and the subroutines together. However, HELLO.FTN is so simple that binding is not necessary. But for demonstration purposes, execute the following command line:

```
$ BIND hello.bin -BINARY greetings <RET>
```

The “-BINARY greetings” option instructs the binder to create an executable program called “greetings”. You must specify -BINARY and some filename, or the binding won’t work.

If there were no errors, the binder returns the following message: “All Globals Resolved.”

Now you’re ready to run the program. If you bound the programs, the following command line will execute it:

```
$ greetings <RET>
```

The program’s message will appear on your screen.

If you did not bind the program, execute it by typing

```
$ hello.bin <RET>
```

Although we used FORTRAN to illustrate program development, the steps for creating Pascal and C programs are similar. If HELLO were a Pascal source file, that is, HELLO.PAS, the compile command line would be

```
$ PAS hello.pas <RET>
```

If it were written in C, the following command line would compile HELLO.C:

```
$ CC hello.c <RET>
```

Binding is the same, regardless of the program’s language.

What we’ve provided is simply a sketch of the program development process. For details on the compilers and their options, refer to the appropriate language manual: the *FORTRAN User’s Guide*, the *Pascal User’s Guide*, or the *C User’s Guide*.

The *Language Level Debugger Manual* describes how to use DEBUG to detect and correct errors in your programs.

Printing A File

In order to print a file, your network must support a printing device, and it must execute a process called the Print Server. When the Print Server is running, you can print any file using the Shell command PRF (PRINT FILE). For example, the following command prints the file Sample_Edit on the line printer:

```
$ PRF Sample_Edit <RET>
```

To print the file on another printer, use the `-pr` option to specify the printer name. For example,

```
$ PRF Sample_Edit -pr spin <RET>
```

Prints the same file on a printer called `spin`. The system returns the following message:

```
"/node_name/sample_edit" copied to "/sys/print/spin.b.you.sample_edit."
```

This means the PRF command has copied the file `Sample_Edit` to a temporary file called `"spin.b.you.sample_edit."` This file resides on the node that controls your printer, in the `/Sys/Print` directory. Because PRF has submitted a *copy* of your file to the printer, you can edit the original version without changing the copy that is printing.

Ask your System Administrator for the names of the printing devices at your installation.

Chapter 4

Using the Display Manager (DM)

The Display Manager (DM) lets you issue commands to create and control windows and processes, read and edit files, and load and display character fonts. There are four ways to issue a DM command:

- Type the command name in the DM's input pad
- Press a specially-defined control character (CTRL/key) sequence
- Press a specially-defined (and labeled) function key
- Define your own function key

The *DOMAIN System Command Reference Manual* describes every DM command. This chapter concentrates on frequently-used control character sequences and function keys.

We explain the standard key definitions that were supplied with your system. After you become an experienced DM user, you may want to change the definitions of certain function keys. There are two ways to do this. Every time you press a function key, the DM reads a key definitions file (in the /SYS/DM directory) to determine what action it should carry out. You can change this file using <EDIT>. To redefine a few individual keys, use the DM command KD. Read the *DOMAIN System Command Reference Manual* when you're ready to experiment with key definitions.

Background Information

In order to understand how the DM works, it is important that you understand windows and pads. Earlier chapters introduced these concepts. Let's take a closer look.

Windows are pretty much what you think they are: openings on the screen through which you view information stored in the system. **Pads** are two-dimensional containers which hold the information that you display.

Windows have such attributes as size and position on the screen, position over a pad, scrolling modes, and possibly color. You can imagine that they are pieces of paper that you can stack and shuffle around on the screen just as you would if they were on your desk. Many of the DM commands which follow allow you to manipulate windows. There are three kinds of pads: input, transcript, and edit pads. **Input** pads accept keyboard input. Programs that read input pads process one line at a time in sequential order. As you may have noticed from examples that you have tried in earlier chapters, input pads are temporary. The DM deletes them when you terminate the process that reads them.

Transcript pads keep a running record (or "transcript") of your interaction with programs by displaying your input and the program's output. Transcript pads are temporary, read-only pads. You cannot modify transcript pad contents, and the DM deletes the transcript pad when you close the window or log off.

Edit pads contain copies of files that the DM displays when you press <READ> or <EDIT>. You can write to pads created with <EDIT>. Pads created with <READ> are read-only. (If you want to write to a pad created using <READ>, press CTRL/M. This saves you the trouble of closing the read-only pad and reopening it with <EDIT>.)

Later in this chapter you'll learn how to use the DM to copy text from read-only pads and paste it into pads that are not read-only (input pads and pads created with <EDIT>). DM cut and paste commands are extremely useful for a variety of tasks, as you will see.

And now to begin.

Moving the Cursor

The most basic operation you can perform is to move the cursor. You position the cursor on the screen by using the touchpad (if present), the arrow keys, or one of the other special cursor keys described below.

Task	Pre-Defined Key
Move left one character	←
Move right one character	→
Move up one line	↑
Move down one line	↓
Move to beginning of line	⌞
Move to end of line	⌟
Tab right	<TAB>
Tab left	CTRL/<TAB>
Move to DM Input Pad	<CMD>
Move to next window on screen	<NEXT WNDW>

Using the Arrow Keys

The four arrow keys located at the left of the keyboard move the cursor one character or line in the indicated direction. These keys also have an auto-repeat feature, so that you can move quickly across the screen by holding a key down.

The two arrow keys that end in a vertical bar (⌞ →⌟) cause the cursor to move to the extreme ends of the current line, to either the first or the last character.

Using <TAB>

<TAB> causes the cursor to move to the right until it hits the next tab stop. Please note that an ASCII tab character is NOT inserted; this key only positions the cursor. Pressing <CTRL> and <TAB> simultaneously causes the cursor to tab left.

The first tab stop is five spaces from the left side of the window. Subsequent tab stops are set every four spaces. To change tab settings, use the DM command TS (TAB__SET, described in the *DOMAIN System Command Reference Manual*).

Using <CMD> and <NEXT WNDW>

<CMD> automatically places the cursor in the DM input window so that you may issue a DM command. When the DM completes that command, it returns the cursor to wherever it was before you pressed <CMD>.

<NEXT WNDW> places the cursor in the next *fully unobscured* window on the screen. Don't confuse <NEXT WNDW> with <POP> or CTRL/P. <POP> and CTRL/P bring an *obscured* window into full view. <NEXT WNDW> simply moves the cursor into another window that is already completely visible.

Defining Points and Regions

Now that you can move the cursor, let's talk about points and regions. Some DM commands (and the keys which invoke them) require that you "point" with the cursor or define a region on the screen so that the DM will understand where it is that you want it to carry out the command. To point, simply move the cursor to the desired place. For example, to point to a window, place the cursor anywhere inside the window. The window control keys you press (which we'll discuss in a minute) read the cursor position to determine which window you mean. Other DM function keys also perform their tasks at whatever point the cursor is currently located.

Some commands require a "region" in which to work. (Those that do are indicated in the descriptions which follow.) The cut and paste editing functions, for instance, need to know which blocks of text to cut and paste. A region for DM operations is simply the area between two points. To declare a region point the cursor to the start of the region, press <MARK>, then point to the end of the region and issue the DM command in question. That's all there is to it. For those commands that require the declaration of some region (the window creation command, for instance, which requires a window size and location), defaults apply if you do not specify a region. The *DOMAIN System Command Reference Manual* describes the defaults that apply to each DM command. The default regions will work fine for our purposes here.

Creating a Window: Read and Edit Pads

Before you can read or edit a file, you must first create a pad to hold it and a window through which to view it. The commands described below perform those functions.

Task	Pre-Defined Key
Create edit pad and window in which to view it	<EDIT>
Create view window (read-only pad)	<READ>

Creating an Edit Pad

Pressing <EDIT> causes the DM to move the cursor to the DM input window and issue an "Edit file:" prompt. Type the name of the file that you wish to edit. If the file you specify does not exist, the DM creates a pad with the specified name. (Notice that we use <EDIT> rather than the Shell command CRF (CREATE__FILE) to create a new file. The CRF command creates an empty, permanent file intended primarily for debugging purposes.)

If you respond to the DM's "Edit file:" prompt with the name of a file that does exist, the DM opens it for editing. The DM creates a copy of the original file. The name of this copy consists of the original filename and a ".BAK" suffix. (See Appendix C for a list of standard name suffixes.) When you finish editing, the DM copies the edit pad to the original file. The .BAK (backup) file saves the pre-edited version of the file until you edit the file again, or explicitly delete the backup file.

After the DM creates an edit pad, it opens a window to it. You may then use other DM commands to manipulate text in the pad. See the "Editing a Pad" section later in this chapter.

To close an edit pad and its window, use <EXIT> or CTRL/Y (to save the new text), or <ABORT> or CTRL/N (to ignore any changes). See "Closing a Window" later in this chapter for details on these keys.

Creating a Read-Only Pad

Pressing <READ> causes the DM to move the cursor to the DM input window and issue a "Read file:" prompt. Type the name of the file that you wish to read. This file must exist; if it does not, the system will respond "File not found" in the DM output window. When the file is found, the DM opens it in "read-only" mode, and creates a window to view it.

You cannot change files accessed with <READ>. If you try to insert or delete material from read-only files, the DM will remind you that the "Text is read-only". (If you are authorized to edit the file, press CTRL/M. CTRL/M instructs the DM to switch from read-only to edit mode, and makes the file available for editing. Chapter 6 explains how the system protects files from unauthorized use.)

To close a read-only edit pad and its window, use <ABORT> or CTRL/N. See "Closing a Window" later in this chapter for details on those keys.

Managing Windows

The window control keys change the size, position, and characteristics of windows on the screen. In addition, all window control keys cause the DM to completely display the selected window if any part of it is hidden. Do not confuse these keys with pad control keys, which move a pad around under a particular window.

<F8> and <AGAIN> provide an easy way to repeat one-line Shell commands and pathnames. You can copy a line from a transcript pad into the Shell's process input window using this key.

Task	Pre-Defined Key	
	DN300 Keyboard	Other Keyboards
Enlarge or reduce a window Move a window Shuffle a window	<GROW> <MOVE> <POP>	CTRL/G CTRL/W CTRL/P
Close window, pad; update file Close window, pad; no update	<EXIT> <ABORT>	CTRL/Y CTRL/N
Hold or release the display	<HOLD>	<HOLD/GO>
Copy text to process input window	<AGAIN>	<F8>

Changing Window Size

You can change the size of a window on the screen by using the <GROW> key (on DN300 keyboards) or the CTRL/G sequence (on other keyboards). To be precise, <GROW> works by moving one edge or a corner of a window across the screen while leaving the other edges and/or corners where they are.

To enlarge or reduce a window, first mark the edge or corner you want to move by positioning the cursor at that edge or corner and pressing <MARK>. Then move the cursor to the edge or corner's new location and press <GROW> or CTRL/G. The marked edge or corner moves to the new cursor position, and the window shrinks or grows accordingly. If you want to move only an edge, move the cursor only in the direction perpendicular to that edge. Moving the cursor vertically and horizontally causes a corner to move.

Moving a Window

The DM operation that moves windows is similar to the one that changes window size, except that only window position — not size — changes. Place the cursor on one edge or corner of the window, press <MARK>, then move the cursor to the edge or corner's desired location and press <MOVE> or CTRL/W. The marked edge or corner moves, and pulls the rest of the window along with it.

Pushing or Popping a Window

This function allows you to display windows that are partially or completely hidden by other windows on your screen. Pressing <POP> (on DN300 keyboards) or the CTRL/P sequence (on other keyboards) pops a window to the top of the pile or pushes a window to the bottom of the pile of windows on the screen. If the cursor rests in a partially obscured window, these keys pop that window to the top of the pile. If the cursor rests in a completely visible window, they push that window to the bottom of the pile.

Closing a Window

The window closing operation normally closes a window and the associated pad. The DM leaves a pad open if there are other windows viewing it, or if it is the last window to an active process. To close a process output (transcript) window, you must first stop the process (see "Stopping a Shell Process" later in this chapter).

To save the changes you've made to an edit pad use <EXIT> or CTRL/Y to close the window. To discard edits you've made to the pad (if any), press <ABORT> or CTRL/N. The window closes and the pad disappears, leaving the original file unchanged.

If you are closing the window to a read-only pad, CTRL/Y and CTRL/N (or <EXIT> and <ABORT>) perform the same operation.

Task	Pre-Defined Key	
	DN300 Keyboard	Other Keyboards
Close window, pad; update file	<EXIT>	CTRL/Y
Close window, pad; ignore changes	<ABORT>	CTRL/N

Once a window is closed, the DM positions the cursor at its most recent position in the previous window.

Setting Hold Mode

<HOLD> (on DN300 keyboards) and <HOLD/GO> (on other keyboards) holds or releases the display within the current window. When you press this key the DM “freezes” the position of the transcript pad. The transcript pad will not display new program output until you press the key again and release the pad. When you release the display, the window automatically displays the end of the transcript pad and any new program output.

Initially, windows are not in hold mode. The window legend contains an “H” when the window is in hold mode.

Copying Text to the Process Input Window

Pressing <AGAIN> (on DN300 keyboards) and <F8> (on other keyboards) copies all the text to the right of the current cursor position into the process input window. This provides a quick way to re-enter a command line that you typed earlier; simply position the cursor at the beginning of the line to be copied (stored in the transcript pad or anywhere else on the screen) and press <AGAIN> or <F8>. If there is anything already in the process input window waiting for processing, the copied line will be appended to that text.

Moving a Pad Under a Window



The following pad control keys move a pad around under a window. Note that these keys are different from window control keys (see “Managing Windows” earlier in this chapter), which move windows around on the screen.

Task	Pre-Defined Key	
	DN300 Keyboard	Other Keyboards
Move cursor to first character in pad	CTRL/T	CTRL/T
Move cursor to last character in pad	CTRL/B	CTRL/B
Move pad by pages	<div style="text-align: center;"> ↑ ↓ </div>	<div style="text-align: center;"> ↓ ↑ </div>
Move pad by lines	<SHIFT>/↑ <SHIFT>/↓	F2 F3
Move pad by characters	<SHIFT>/← <SHIFT>/→	<div style="text-align: center;"> ← → </div>

Moving to the Top and Bottom of a Pad

The CTRL/T sequence causes the DM to position the cursor at the top of the current pad; in other words, on the first character in the pad. CTRL/B performs the complementary function of positioning the cursor at the bottom of the pad, at the last character.

Moving (Scrolling) Vertically by Pages or Lines

The vertical scrolling keys  and  scroll a pad vertically under a window in units of half-pages. A page is defined as the smaller of either of the following values:



- The number of lines that fit in the window
- The number of lines between the bottom of the window and the next form feed (the invisible control character that signals the start of a new page) or frame

This scrolling factor of half-pages can be modified if you like. See the description of the DM command PP (PAD_PAGE) in the *DOMAIN System Command Reference Manual*.

You may imagine that they cause a pad to move under a window or that they cause a window to move over a pad. Recent human factors research seems to indicate that most people favor the latter orientation, hence the boxed arrow keys are reversed on newer keyboards.

The other way to scroll the pad vertically is by lines rather than by pages. The <SHIFT>/↑ and <SHIFT>/↓ sequences on DN300 keyboards, and the F2 and F3 keys on other keyboards move the window up and down one line, respectively. The cursor position does not change relative to the pad; the pad simply slides by under the window.

Moving (Scrolling) Horizontally by Characters

The left and right scrolling keys <SHIFT>/← and <SHIFT>/→ on DN300 keyboards scroll a window over a pad in 5-character increments. The  and  keys on other keyboards scroll a pad under a window in 10-character increments. See the preceding section for an explanation of the reversal.

Editing a Pad

You can use many of the editing functions described below in any pad at any time (provided there's a window open into it) — even those pads that are in read-only mode. In the latter case, of course, those keys that alter the text in some way won't work, but you can still copy text into paste buffers, search for strings, etc. By default, you are always in “insert-mode”: the DM will attempt to insert typed characters into a pad at the current cursor position. You'll get an error message, of course, if the cursor is in a read-only pad.

Since many of the editing operations either require you to define a range of text in which to operate, or to use a “regular expression” to describe a string of characters, we touch on these concepts first. The *DOMAIN System Command Reference Manual* discusses these topics more fully.

Task	Pre-Defined Key	
	DN300 Keyboard	Other Keyboards
Set read/write mode Set insert/overstrike mode	CTRL/M <INS>	CTRL/M <INS MODE>
Insert NEWLINE character Insert new line after current line Insert end-of-file mark	<RETURN> <F1> CTRL/Z	<RETURN> <F1> CTRL/Z
Delete character at cursor Delete character before cursor Delete "word" of text Delete from cursor to end of line Delete entire line	<CHAR DEL> <BACK SPACE> <F6> <EOL> <LINE DEL>	<CHAR DEL> <BACK SPACE> <F6> <F7> <LINE DEL>
Copy text to paste buffer Cut (delete) text and write it to paste buffer Paste (write) text in paste buffer into pad	<COPY> <CUT> <PASTE>	CTRL/C CTRL/E CTRL/O
Search forward for string Repeat last search forward Search backward for string Repeat last search backward Abort search Set case comparison for search	<CMD> /string/ CTRL/R <CMD> \string\ CTRL/U CTRL/X <CMD> SC [-ON -OFF]	<CMD> /string/ CTRL/R <CMD> \string\ CTRL/U CTRL/X
Substitute string2 for all occurrences of string1 in defined range Substitute string2 for first occurrence of string1 in each line of defined range	<CMD> S/string1/string2/ <CMD> SO/string1/string2/	
Undo previous command(s)	<UNDO>	none
Update edit file without closing edit pad	<SAVE>	none

Defining a Range of Text

The text editing commands that perform cut, paste, and substitute functions operate on a range of text. You mark that range just as you would mark any other region in a pad (see the "Defining Points and Regions" section earlier in this chapter); i.e., you place the cursor at the start of the range, press <MARK>, then move the cursor to the end of the range, press <CMD>, and issue the command in question. Please note that the character under the cursor at the end of the range is NOT included within the range.

If you do not define a range, the default range for cut, paste, and substitute commands begins at the current cursor position and ends at the end of the line (this includes the NEWLINE character that you inserted by pressing <RETURN>).

Setting Read/Write, Insert/Overstrike Modes

All edit pads are controlled by a very important feature of the DM: the “modes” in which the DM currently operates. The modes determine, for instance, whether or not you can make changes to the material in the pad. The following keys allow you to manipulate the mode settings.

CTRL/M — Set read/write mode

The CTRL/M sequence on both keyboard models moves a pad into or out of read-only mode: it switches the current setting. The pad must be in write mode in order for you to insert or delete anything.

An “R” appears in the window legend of a pad in read-only mode. The “R” disappears in write mode.

An edit pad which has been modified cannot be made read-only without first copying the changes to a disk file using the DM command PW.

<INS> or <INS MODE> — Set insert/overstrike mode

These keys (<INS> on DN300 and <INS MODE> on other keyboards) move the current pad into or out of insert mode: they switch the current setting. In insert mode, characters you type are inserted into the pad without replacing or overstriking any existing characters. This causes existing text to shift to the right as new text is added. In overstrike mode (i.e., insert mode turned off), characters typed at the keyboard *replace* those under the cursor. This can be useful for entering information into pre-formatted files without disturbing the format.

An “I” appears in the window legend of a pad that is in insert mode. The “I” disappears in overstrike mode.

Inserting Text

As we have said, any pad that is in write mode automatically accepts anything typed at the keyboard as input to that pad. The keys below perform special insertion functions.

<RETURN> — Insert NEWLINE

<RETURN> inserts (or overstrikes, depending on the current mode) a NEWLINE character at the current cursor position.

<F1> — Insert new line

Pressing <F1> on both DN300 and other keyboards causes the Display Manager to insert a blank line following the line containing the cursor. The position of the cursor within the original line is insignificant.

CTRL/Z — Insert end-of-file mark

The CTRL/Z sequence on both DN300 and other keyboards causes the DM to insert a stream end-of-file mark (EOF) in the pad. If the line containing the cursor is empty, the end-of-file mark is written on that line. Otherwise, the end-of-file mark is inserted following the current line. Inserting EOF into a Shell process input pad signals the DM to close that pad. Windows into a pad can then be closed once the pad itself is closed (see “Closing a Window” section earlier in this chapter).

Deleting Text

The following keys perform special deleting functions. Unless otherwise noted, the keys are identical on all our keyboards.

<CHAR DEL> — Delete character at cursor

<CHAR DEL> deletes the character under the cursor. If the character is a NEWLINE, <CHAR DEL> joins two lines.

<BACK SPACE> — Delete character before cursor

If the window is in insert mode, <BACK SPACE> deletes the character preceding the cursor. If the window is in overstrike mode, <BACK SPACE> replaces the preceding character with a blank.

<F6> — Delete “word” of text

<F6> has been pre-defined to delete a “word” of text starting at the current cursor position. A “word” in this case consists of a string of lowercase letters. The deletion stops at the next capital letter, punctuation mark, or space encountered. Deleted text is written to the default paste buffer for re-insertion elsewhere if you so desire. (See the next section “Cutting and Pasting Text”).

<EOL> and <F7> — Delete from cursor to end of line

<EOL> on DN300 keyboards and <F7> on other keyboards have been pre-defined to delete text from the current cursor position to the end of the current line (excluding the NEWLINE character). The DM writes deleted text to a paste buffer so you can re-insert the text elsewhere if you so desire (see the next section “Cutting and Pasting Text”). (To re-insert text you deleted by pressing <EOL>, press <SHIFT>/<EOL>. To re-insert text you deleted by pressing <F7>, press either CTRL/O or <PASTE>.)

<LINE DEL> — Delete entire line

<LINE DEL> deletes the entire line containing the cursor. The position of the cursor within the line is insignificant. The DM writes the deleted text to a paste buffer so you can re-insert it elsewhere if you so desire. (See the next section.) (On DN300 keyboards you can re-insert this text by pressing <SHIFT>/<LINE DEL>. On other keyboards, you can re-insert this text by pressing either CTRL/O.)

Cutting and Pasting Text

The following keys perform cut and paste operations. Cutting and pasting involves moving blocks of text from one place to another in a pad (or between pads). To accomplish this, the DM uses “paste buffers”: temporary files that hold text you have copied or cut so that it can be pasted in elsewhere. There can be many paste buffers (up to a hundred, in fact), each containing different blocks of text. You distinguish between them by giving them names when you create them. If you don’t want to bother with all those buffers, however, you can simply use the default (unnamed) buffer associated with <CUT>, <COPY>, and <PASTE>. Making use of the other multiple named buffers requires you to use literal DM commands, which we are leaving to the *DOMAIN System Command Reference Manual* to discuss.

<COPY> and CTRL/C — Copy text to paste buffer

<COPY> on DN300 keyboards and the CTRL/C sequence on other keyboards copy a range of text from any pad into the default paste buffer. The copied text remains undisturbed. Define the range to be copied as described in “Defining a Range of Text” earlier in this chapter.

<CUT> and CTRL/E — Cut (delete) text and write to paste buffer

<CUT> on DN300 keyboards and the CTRL/E sequence on other keyboards copy a range of text into the default paste buffer, then delete the original text from the pad. These keys can be used only in a writable pad. Define the range to be copied as described in “Defining a Range of Text” earlier in this chapter.

<PASTE> and CTRL/O — Paste (write) text in buffer into pad

<PASTE> on DN300 keyboards and the CTRL/O sequence on other keyboards insert the contents of the default paste buffer into a pad at the current cursor position. The paste buffer contents are unchanged by this operation, so you can make repeated insertions throughout the pad. These keys can be used only in a writable pad. You can use cut and paste commands to move text from one window to another.

Searching for Text

There are no pre-defined keys provided to perform initial searches for strings of characters (words), although you can repeat searches using control/key sequences as described below. Initial searching is accomplished by moving the cursor to the DM input window (using <CMD>) and issuing the search commands “/string/” and “\string\”. These commands are identical for DN300 keyboards and other keyboards.

Actually, the search and substitute operations described here are much more powerful than presented. You can use a “regular expression” to define the string of text for which you are searching. Regular expressions allow you to describe general character strings without worrying about their exact contents. See “Using Regular Expressions” in the *DOMAIN System Command Reference Manual*.

/string/ and \string\ — Search forward and backward for string

The search operation moves the cursor to the beginning of the text that matches the string of characters you specify. A string enclosed in slashes (/string/) causes a forward search, and a string enclosed in backslashes (\string\) causes a backward search.

If the search fails, the DM reports “No match.”

Searches are case insensitive by default. This means that /mary/ would locate the word “MARY.” Use the SC command (below) to perform case-sensitive searches.

CTRL/R and CTRL/U — Repeating forward and backward searches

The control key sequences CTRL/R and CTRL/U repeat the last search forward or backward, respectively. The DM saves the most recent search instruction, so you may repeat it even if you have issued other (non-searching) commands since then.

CTRL/X — Abort search

The CTRL/X sequence aborts a search in progress. The DM returns the message “Search aborted.” It does not move the window.

SC — Set case comparison for search

A search can be either case-sensitive or case-insensitive. In case-sensitive searches, the characters must match in case. In case-insensitive searches, uppercase and lowercase letters are considered equivalent. By default, searches are case-insensitive.

The SC command controls search case-sensitivity for all windows on the screen. (You cannot have different settings for different windows.)

Position the cursor in the DM input window by pressing <CMD>, then type the SC command. The -ON option explicitly specifies a case-sensitive search; the -OFF option explicitly specifies a case-insensitive search. The SC command without options switches the current case comparison setting. For example,

Command: SC -ON <RET> (enable case sensitivity for all windows)

NOTE: The SC command has no affect on *substitution* operations, only *search* operations. Substitutions are always sensitive to the case of the strings involved.

Substituting Text

Like text search operations, text substitute operations have no pre-defined keys. You must position the cursor in the DM input window by pressing <CMD>, then issue the substitution command.

S/string1/string2/ — Substitute all occurrences of matched string

The S command substitutes one string of characters (“string2”) for another (“string1”) over a defined text range (see the “Defining a Range of Text” section earlier in this chapter). The command does not move the cursor or the window, but does update the window when the substitution is completed. Strings used with this command are also saved for use in later substitution commands.

All substitutions are case sensitive, unlike searches, which ignore case unless told otherwise. Substitution case sensitivity cannot be disabled.

It is possible to repeat substitutions and use default values for “string1” and “string2”, but this can be risky unless you know what you are doing. See the S command description in the *DOMAIN System Command Reference Manual* for details.

EXAMPLE: <CMD> S/mary/Mary Sue/ (This substitutes “Mary Sue” for all occurrences of “mary” in whatever text range has been marked.)

SO/string1/string2/ — Substitute first occurrence of matched string

The SO command is identical to the S command, except that “string2” replaces only the first occurrence of “string1” in each line of the defined range.

Undoing Previous Commands

<UNDO> works by compiling a history of activities in input and edit pads in reverse chronological order. Pressing <UNDO> reverses the effect of the most recent DM command. Successive use of <UNDO> will undo further back in history. Please note that <UNDO> only works for DM operations; it will not reverse the effect of Shell commands once those commands have been executed since the DM does not keep track of what the command Shell is doing.

The UNDO buffers (one edit undo buffer per edit pad and one input undo buffer per input pad) are circular lists that, when full, eliminate the oldest entries to make room for new ones. Entries are grouped together in sets. For example, a S (SUBSTITUTE) command may change 5 lines. While UNDO considers this to be five entries, the five entries are grouped into a single set so that one UNDO will change all five lines back to their original state. When a buffer becomes full, the oldest *set* of entries is erased. This means that UNDO will never partially undo an operation: it will either completely undo it or do nothing.

If <UNDO> is not available on your keyboard, you can achieve the same effect by pressing <CMD> and then typing “undo”.

Updating an Edit File

<SAVE> updates a file that is being edited. It is valid only for writable edit pads. The first time you press <SAVE>, the DM writes the contents of the edit pad to the file that is being edited, without closing the edit pad. The previous contents of the file are saved in a file with the same name and the added suffix .BAK. Subsequent uses of <SAVE> rewrite the new file and leave the .BAK version of the file unchanged.

Creating a Process

The DM controls not only the windows and pads associated with your screen, but also the creation and destruction of command Shell environments in which you do all those other things commonly associated with using a computer (writing and running your own programs, for instance).

You can create a new Shell process, along with all the necessary pads and windows, by pressing <SHELL>. This automatically runs the process creation program, creates input and transcript pads, and opens windows into those pads. The DM places the cursor in the Shell process input window (denoted by the “\$” prompt), and the process is ready to receive Shell commands (see Chapter 5).

Task	Pre-Defined Key
Create new process, transcript pad, and windows	<SHELL>

Stopping a Process

Process termination keys stop a Shell process or stop another executing program.

Task	Pre-Defined Key
Stop a program	CTRL/Q
Stop Shell process, close pads and windows	CTRL/Z, CTRL/N

Stopping a Shell Process



You can stop a Shell process, along with all of its input and output pads and windows, by first typing CTRL/Z in the Shell process input window, then, after a moment, CTRL/N. CTRL/Z inserts an end-of-file mark into the input stream, indicating to the DM that there will be no more input for this Shell process. The input window disappears, and the message “***Pad Closed***” appears in the transcript window. CTRL/N then deletes that output window, and repositions the cursor in the previous window. Wait for the “***Pad Closed***” message before typing CTRL/N. Otherwise you will get a message saying that the window is still active (since the pad isn’t closed yet). CTRL/N will work as soon as the DM closes the pad.

Stopping a Program

Typing CTRL/Q while a program is running causes the DM to bring the program to a halt. If the DM cannot terminate the program in an orderly manner, CTRL/Q fails and the program continues to run. To force programs to stop, use the DM command DQ. See the *DOMAIN System Command Reference Manual* for details.

Display Manager (DM) Command Summary

MOVING THE CURSOR

Task	DM Command	Pre-Defined Key	
		DN300 Keyboard	Other Keyboards
Move left one character Move right one character Move up one line Move down one line Set arrow key scale factors (raster units)	AL AR AU AD AS x y	← → ↑ ↓ none	← → ↑ ↓ none
Move to start of next line Move to beginning of line Move to end of line Move to top line in window Move to bottom line in window	AD;TL TL TR TT TB	CTRL/K ← → <SHIFT>/  <SHIFT>/ 	CTRL/K ← → none none
Tab left Tab right Set tabs	THL TH TS [n1 n2 ...] [-R]	CTRL/<TAB> <TAB> none	CTRL/<TAB> <TAB> none
Move to DM input pad Move to next window on screen Move to previous window Move to next window in which input is enabled	TDM TN TLW TI	<CMD> <NEXT WNDW> CTRL/L none	<CMD> <NEXT WNDW> CTRL/L none

CREATING A PROCESS

Task	DM Command	Pre-Defined Key
Create new process, transcript pad, and windows	CP pathname	<SHELL>
Create new process without transcript pad or windows	CPO pathname	none
Create server process	CPS pathname	none







CONTROLLING A PROCESS

Task	DM Command	Pre-Defined Key
Quit, stop, or blast a process	DQ [-S -B]	CTRL/Q
Suspend execution of a process	DS	none
Resume execution of suspended process	DC	none

CREATING A WINDOW TO READ AND EDIT PADS

Task	DM Command	Pre-Defined Key
Create edit pad and window in which to view it	CE pathname	<EDIT>
Create view window (read-only pad)	CV pathname	<READ>
Create copy of existing window	CC	none

MOVING A PAD UNDER A WINDOW

Task	DM Command	Pre-Defined Key	
		DN300 Keyboard	Other Keyboards
Move top of pad into window Move cursor to first character in pad	PT PT;TT;TL	none CTRL/T	none CTRL/T
Move bottom of pad into window Move cursor to last character in pad	PB PB;TB;TR	none CTRL/B	none CTRL/B
Move pad n pages Move pad n lines	PP [+/-]n PV [+/-]n	  <SHIFT>/ ↑ <SHIFT>/ ↓	  F2, F3 F2, F3
Move pad n characters	PH [+/-]n	<SHIFT>/ ← <SHIFT>/ →	 
Save pad in 'pathname'	PN pathname	none	none

MANAGING WINDOWS

Task	DM Command	Pre-Defined Key	
		DN300 Keyboard	Other Keyboards
Grow or shrink a window Move a window Push or pop a window on stack	WG WM WP	<GROW> <MOVE> <POP>	CTRL/G CTRL/W CTRL/P
Close (delete) a window Close window, pad; update file Close window, pad; no update	WC [-Q -F] PW;WC -Q WC -Q	none <EXIT> <ABORT>	none CTRL/Y CTRL/N
Set scroll mode Set auto-hold mode Toggle scroll and autohold modes Set hold mode	WS [-ON -OFF] WA [-ON -OFF] WA;WS WH [-ON -OFF]	CTRL/S none CTRL/A <HOLD>	CTRL/S none CTRL/A <HOLD/GO>
Define position of default window 'n'	WDF n	none	none
View latest output in Shell transcript pad Copy text to process input pad	AU;AU;PB;TI DR;TR;XC;TL;TI;TB; TR;XP;TR	CTRL/V <AGAIN>	CTRL/V <F8>
Copy text to DM input pad for use with <READ>	DR; /[~a-z0-9\$@/_@~._`~`]/ XC CV__FILE;TDM; ES 'CV ';XP CV__FILE; TR;EN	<SHIFT>/<READ>	none

MANAGING THE DISPLAY ENVIRONMENT

Task	DM Command	Pre-Defined Key	
		DN300 Keyboard	Other Keyboards
Request help	none	<HELP>	none
Log on Log off Shut down system	L id[proj[org]][-P][[-H] LO [-F] SHUT [-F]	none none none	none none none
Place a mark Go to a mark Clear mark stack	DR GM CMS	<MARK> <SHIFT>/<MARK> none	<MARK> none none
Display cursor coordinates (line, column; x/y coordinates)	=	none	none
Acknowledge DM alarm Acknowledge alarm and pop window	AA AP	none none	none none
Set background color Set window color	BGC [-ON -OFF] INV [-ON -OFF]	none none	none none
Refresh entire screen Refresh window	RS RW [-R]	CTRL/F none	CTRL/F none
Load font for use in pads	FL pathname	none	none
Declare keyboard type	KBD [type]	none	none

EDITING A PAD

Task	DM Command	Pre-Defined Key	
		DN300 Keyboard	Other Keyboards
Set read/write mode Set insert/overstrike mode	RO [-ON -OFF] EI [-ON -OFF]	CTRL/M <INS>	CTRL/M <INS MODE>
Insert string Insert NEWLINE character Insert new line after current line Insert raw (noecho) character Insert end-of-file mark	ES 'string' EN TR;EN;TL ER nn EEF	[Default DM operation] <RETURN> <F1> none CTRL/Z	<RETURN> <F1> none CTRL/Z
Delete character at cursor Delete character before cursor Delete "word" of text Delete from cursor to end of line Delete entire line	ED EE DR;/[~A-Z0-9\$__]XD ES ' ';EE;DR;TR;XD;TL;TR CMS;TL;XD	<CHAR DEL> <BACK SPACE> <F6> <EOL> <LINE DEL>	<CHAR DEL> <BACK SPACE> <F6> <F7> <LINE DEL>
Copy text to paste buffer Cut (delete) text and write it to paste buffer Paste (write) text in paste buffer into pad	XC [name] XD [name] XP [name]	<COPY> <CUT> <PASTE>	CTRL/C CTRL/E CTRL/O
Search forward for string Repeat last forward search Search backward for string Repeat last backward search Abort search Set case comparison for search	/string/ // \string\ \\ SQ SC [-ON -OFF]	none CTRL/R none CTRL/U CTRL/X none	none CTRL/R none CTRL/U CTRL/X none
Substitute string2 for all occurrences of string1 in defined range Substitute string2 for first occurrence of string1 in each line of defined range	S/string1/string2/ SO/string1/string2/	none none	none none
Undo previous command(s)	UNDO	<UNDO>	none
Update edit file without closing edit pad	PW	<SAVE>	none

Chapter 5

Using the Shell

The Shell is a program that invokes utility programs, such as the delete file program (DLF) or the change name program (CHN). The *DOMAIN System Command Reference Manual* describes all of the programs you can invoke using Shell commands. This chapter explains the environment in which the Shell processes your commands. As you read the following pages, you'll discover how the Shell interprets your input and how it locates the command file (program) you request. You'll learn how to use symbols that have special meaning to the Shell, and how to create your own Shell commands, called Shell **scripts**.

Command Line Processing

Most command lines consist of a command name followed by arguments to the command. For example,

```
$ COMMAND arg1 arg2 arg3 ...argn <RET>
```

Notice that spaces separate the command from its arguments, and separate arguments from each other. When you press <RETURN> the Shell searches for a file with the same name as the command you've specified. If the Shell finds such a file, it loads and executes it as a program. Next, the Shell passes any arguments you included on your command line to this program. After it executes the program, the Shell returns the "\$" prompt. This means it is ready for your next command.

Command Search Rules

Command search rules determine which directories the Shell searches to locate commands and programs. You can set or show command search rules using the Shell command CSR (COMMAND__SEARCH__RULES). By default, the Shell searches for command files in the following three directories:

1. Your working directory (".").
2. Your personal command directory, ~ COM.
3. The system command directory /COM.

As soon as the Shell finds a name that matches the one requested, it attempts to execute the program.

NOTE: Don't create text files with the same names as commands, or the Shell will attempt to execute the text file as if it were a command file.

The ~ COM directory is a subdirectory that you may create in your naming directory. When the Shell does not find the command file you specify in the working directory, it checks this directory. Therefore, ~ COM is a good place to store Shell scripts and other frequently used programs. You are not required to create a ~ COM directory — no error occurs if one does not exist.

The /COM directory contains the command files supplied with the system. (Later in this chapter, you'll learn how to create your own command files.) 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

```
$ SRF Myfile <RET>
```

The Shell looks for the SRF (SORT__FILE) command in your working directory first, and then in ~COM. Finally, it finds the SRF command file in /COM, sorts Myfile, and displays the output. (This example assumes that you have not created an object named SRF in your working directory or in ~COM.)

Usually you refer to commands with relative pathnames (just the command name). Absolute pathnames (such as //Mynode/Myfiles/Mycommand) override command search rules. If you use an absolute pathname, the system searches only the directory you specify — not the working directory, ~COM, and /COM.

Command Names

Shell command names are usually abbreviations of longer descriptive names. Most consist of a verb and a noun, such as COPY__FILE (CPF) and DELETE__FILE (DLF). Some command names (DATE, HELP) consist of just nouns, and imply the verb “set” or “show.” Other command names are verbs that describe the program’s action (BIND, DEBUG). Table 5-1 lists standard abbreviations. Type

```
$ HELP command names <RET>
```

for a complete list of Shell command names and abbreviations.

Table 5-1: Standard Shell Command Abbreviations

Verb	Abbreviation	Noun	Abbreviation
archive	arc	directory	d
catalog	ct	file	f
change	ch	link	l
close	cl	name	n
compare	cm	network	net
copy	cp	node	node
create	cr	object	ob
delete	dl	pattern	pat
dismount	dmt	server	svr
find	f	status	st
initialize	in	stream	str
list	l	tree	t
lock	lk	volume	vol
mount	mt		
print	pr		
read	r		
salvage	sal		
sort	sr		
uncatalog	uct		
unlock	ulk		

Command Format

Most Shell commands accept **command arguments**. A command argument specifies the object upon which the command will act. For example,

```
$ CPF file5 file6 <RET>
```

In this example, file5 and file6 are the command arguments. The CPF (COPY_FILE) command copies file5 to file6. You must use one or more spaces to separate commands from arguments, and to separate arguments from each other. To include blank spaces within an argument, supply the argument in single or double quotation marks.

Most commands allow you to modify their action by specifying one or more **command options**. Precede each option with a hyphen. Do not include any space between the hyphen and the option. For example,

```
$ LD -c <RET>
```

The -c option instructs LD (LIST_DIRECTORY) to list entries in a single column.

You can append the following options to any Shell command:

-HELP	Displays detailed usage information.
-USAGE	Displays a brief usage summary.
-VERSION	Displays software version number.

Check the system help files (type HELP command name) for information about specific command options.

To input more than one command on a single line, separate the commands with a semicolon. For example,

```
$ LD;DATE <RET>
```

This command line lists the contents of the current working directory, then displays the date.

Finally, 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 //Mynode/My@  
$_directory <RET>
```

This command sets the working directory to //Mynode/Mydirectory. The @ sign causes the Shell to ignore the <RETURN> character 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 does not interpret this character when it reads your command.

The @ symbol is an escape character. It suppresses the meaning of the special character it precedes. In the previous example, the @ symbol preceded a <RETURN> character and canceled the <RETURN> character's usual effect. The <RETURN> character usually

delimits a command line. In this case, the @ symbol caused the Shell to ignore the <RETURN> character. The @ symbol and the <RETURN> character both have special meaning. The next section explains how to use other special characters.

Redirecting Input/Output

By default, a process reads input from its input pad, and writes output to its transcript pad. This section explains the special characters you can use to redirect input and output (I/O) during interactive sessions. The following sections describe each character in detail. (See the *DOMAIN System Command Reference Manual* if you're interested in redirecting error I/O.) Table 5-2 lists these characters.

Table 5-2: Special Characters for Interactive Use

Special Character	Function
>filename	Writes output to filename
<filename	Reads input from filename
-	Reads data from the input pad
*	Reads filename arguments from the input pad
* <filename	Reads filename arguments from filename
CMD1 CMD2	Uses the first command's output as the second command's input.

Writing Output to a File

To instruct the process to write output to a file rather than to the transcript pad, use the greater than (>) symbol. Note that one or more spaces must precede the symbol, and that no spaces appear between the symbol and the output filename. For example, the next command writes output to a file named File1.fmt:

```
$ FMT File1 >File1.fmt <RET>
```

FMT (FORMAT TEXT) formats File1 and writes the output (the formatted file) to File1.fmt instead of the transcript pad. The Shell creates the output file if it does not already exist.

Reading Data from a File

To instruct the process to read data from a file rather than the input pad, use the less than (<) symbol. For example, the next command reads data from a file named File1.fmt:

```
$ FPAT <File1.fmt DOMAIN <RET>
```

FPAT (FIND_PATTERN) displays the lines in File1.fmt that contain the string "DOMAIN."

Reading Data from Standard Input

Most commands that accept filename arguments automatically read data from the file you specify. (`PRINT__FILE` is a good example). To instruct such commands to read *data* (not names) from the process input window (standard input), use the dash (-) symbol. For example,

```
$ CATF - >foo <RET>
this is a foo file <RET>
CTRL/Z
```

This command line causes the `CATF` (`CATENATE__FILE`) command to read the text “this is a foo file” from the process input window, and write output to the file called `foo`. The end-of-file character (`CTRL/Z`) instructs `CATF` to stop reading from standard input, and terminates program execution.

```
$ PRF - <RET>
Print this line on the line printer. <RET>
CTRL/Z
```

This command line causes `PRF` (`PRINT__FILE`) to print the line “Print this line on the line printer.”

Some programs can read input from both a list of files and from standard input. For these programs, simply specify the filename “-” to represent standard input. For example,

```
$ FMT File1 - File2 <RET>
```

This command formats `File1`, then formats data received from standard input, and finally formats `File2`.

Reading Names from Standard Input

To instruct commands to read *names* (not data) from standard input, use the asterisk (*) symbol. For example,

```
$ PRF * <RET>
```

causes `PRF` to read filename arguments from the input pad. Therefore, if you type

```
$ PRF * <RET>
File2 <RET>
File3 <RET>
File4 <RET>
CTRL/Z
```

`PRF` reads the names `File2`, `File3`, and `File4` from the input pad, and prints the contents of these files.

To instruct commands to read a list of names from a file, use this format:

```
$ COMMAND * <Filename <RET>
```

This format instructs `COMMAND` to read names from standard input, and redirects standard input to `Filename`. Therefore,

```
$ PRF * <File1 <RET>
```

instructs `PRF` to print the contents of each filename listed in `File1`.

Using Pipes and Filters

If you place two commands on one line, and separate them with a vertical bar, the Shell uses the output of the first command as input to the second command. For example,

```
$ SRF File3 | DLDUPL <RET>
```

`SRF` (`SORT__FILE`) sorts the contents of `File3` and passes the output to the `DLDUPL` (`DELETE__DUPLICATE__LINES`) command. `DLDUPL` strips out duplicate lines. The vertical bar (`|`) between the commands is called a **pipe**.

Commands such as `SRF` and `DLDUPL`, which copy standard input to standard output (making some changes along the way), are called **filters**. A command line that uses pipes and filters is called a **pipeline**. You can use Shell commands or scripts as filters in pipelines.

To use a group of commands as a filter, enclose them in parentheses. Use the following format:

```
$ (COMMAND1; COMMAND2) | COMMAND3 <RET>
```

The Shell uses the output of `COMMAND1` and `COMMAND2` as the input for `COMMAND3`.

Using Wildcards

Wildcards are text strings that you can use to represent one or more pathnames. For example, the wildcard in the following command line matches every file that ends in `.FTN` (FORTRAN programs) in the current working directory:

```
$ LD ?*.FTN <RET>
```

The wildcard `"?"` matches any single character except `<RETURN>`. The wildcard `*` matches zero or more occurrences of the character preceding it.

The next example lists all the objects (files, directories, links) on `Node7`:

```
$ LD //Node7/... <RET>
```

The wildcard `"..."` matches zero or more names subordinate to the starting point. In this example the starting point is `//Node7`.

The next example changes the access control list (ACL) for the file `Ch1`. We use the `EDACL` (`EDIT ACL`) command to grant all users read access to `Ch1`.

```
$ EDACL -a % -read ch1 <RET>
```

The wildcard `"%"` matches zero or more characters (in this case, any user). (Chapter 6 describes access control commands in more detail.)

Most Shell commands that accept pathnames as arguments also accept wildcards. Wildcards are easy to use, but there are a lot of them. This section provides a few examples to introduce this concept to you. See the *DOMAIN System Command Reference Manual* for complete information about wildcards.

Writing Shell Scripts

A Shell script is a file that contains a list of Shell commands. To write a script, simply use <EDIT> to create a file and then insert Shell commands, one per line, then close the file by pressing CTRL/Y. To execute the script, close the edit window and type the script's filename in the Shell input window.

The Shell is a command interpreter and is useful for writing very simple programs. If you want to do something complicated or powerful, use a high-level language, not the Shell.

If you find yourself repeating a sequence of commands over and over again, create a Shell script and insert the commands. Then, you can execute the command sequence with a single command name. For example, to format a file, prepare it for output on the line printer, and print it requires the following three commands:

```
$ FMT File1 >File1.fmt <RET>
$ OS File1.fmt >File1.os <RET>
$ PRF File1.os -npag <RET>
```

The FMT (FORMAT__TEXT) command formats file1 and writes the formatted file to File1.fmt. The OS (OVERSTRIKE) command prepares a formatted file for printing on the line printer, and writes the printable file to File1.os. The PRF (PRINT__FILE) command prints File1.os on the line printer. (Normally PRF begins printing on a new page after 66 lines. In this example, the -npag option causes PRF to ignore standard pagination practice, and use the page breaks supplied in File1.os.)

If you insert the FMT, OS, and PRF command lines in a file called FOP (Format, Overstrike, and Print), you can execute all three commands by typing:

```
$ FOP <RET>
```

To include comments in Shell scripts, precede the comment with the # character. It instructs the Shell to ignore the line.

Substituting Arguments

The Shell script FOP isn't very useful, because it only operates on a single file (File1). To create a more useful script, we need a way to substitute *any* filename for File1. The Shell provides an argument substitution mechanism. To use it, edit the FOP script and change File1 to ^1. Here is the result:

```
$ FMT ^1 >^1.fmt <RET>
$ OS ^1.fmt >^1.os <RET>
$ PRF ^1.os -npag <RET>
```

After you press CTRL/Y and close the edit window, you can execute FOP by typing

```
$ FOP any_filename <RET>
```

The Shell substitutes the first command argument you specify (in this case “any__filename”) for ^1, and then executes the script. So, FOP formats “any__filename” and writes the formatted file to “any__filename.fmt.” Then, FOP overstrikes “any__filename.fmt” and writes the overstruck file to “any__filename.os.”

You can supply up to nine arguments when you invoke a Shell command or script. The Shell numbers the arguments 1 to 9, and uses 0 to refer to the command filename. The Shell replaces any occurrence of ^0 in the Shell script with the script’s filename. It replaces ^n with the nth command line argument. For example, suppose the file MAKE contains the following commands:

```
CATF ^1 ^2 ^3 >^4
PRF ^4
DLF ^4
```

Then, the command line

```
$ MAKE Monday Tuesday Wednesday Temp <RET>
( 0      1          2          3          4 )
```

writes the files Monday, Tuesday, and Wednesday, to a temporary output file called Temp. MAKE prints Temp, and deletes it.

The Shell replaces the ^n sequence with the literal text of the nth argument. If you use quotes to include blank spaces in the argument, the Shell still processes the string as one argument. Consider the contents of a script called LISTDIRS:

```
LD ^1 ^2
```

If we invoke the script with the following command line

```
$ LISTDIRS '-ld -c -dtu' //et/col <RET>
```

The LD command returns the following error message:

```
(ld) Unrecognized or redundant keyword, "-ld -c -dtu"
```

Why? Because the ^1 sequence instructs LD to interpret the options in quotation marks as one argument. LD expects these options as separate arguments.

The Shell provides another type of argument substitution character, the exclamation point (!). It instructs the Shell to break up arguments enclosed in quotation marks that include spaces, and process each option as a separate argument. Therefore, a minor change to LISTDIRS will correct the problem:

```
LD !1 ^2
```

Now, LISTDIRS processes the options enclosed in quotation marks separately. It lists the subdirectories (-ld) of the directory you specify in the second argument, specifies the date each was last used (-dtu), and outputs the information in one column (-c).

As another example, consider the contents of the Shell script FORT shown next. This script compiles and binds a FORTRAN program, allowing you to use different options each time.

```
FTN ^1 -opt !2
BIND ^1.bin !3 -binary ^1
```

In order to compile the FORTRAN source file Prog.ftn with the -DBA and -SAVE options, and then bind it with the -MAP option, invoke FORT with the following command:

```
FORT Prog '-dba -save' '-map >Prog.map'
```

The Shell gathers -dba and -save into one argument for the FORTRAN compiler, and the -map >Prog.map into one argument for the binder. Because we use the ! character to substitute the second and third arguments, the Shell breaks these strings enclosed in quotation marks apart before passing them to FORT. After the Shell substitutes arguments, FORT looks like this:

```
FTN Prog -opt -dba -save
BIND Prog.bin -map >PROG.MAP -binary Prog
```

Use double quotation marks to delimit argument strings that include argument substitution characters (! or ^).

Conditional Statements

You can include conditional statements in your Shell scripts using the Shell command IF. The Shell will test a certain condition, and if the condition is true, execute a second command line. If the condition is not true, the Shell will stop, or execute another command line, depending on your instructions. The following Shell script compiles the Pascal module specified in the first argument (^1) if the second argument (^2) is '-c'. Then the script binds the module with 'library'.

```
IF EQS ^2 '-c' THEN PAS ^1 ENDIF BIND ^1.bin library -b ^1
```

If the second argument is not '-c', or if there is no second argument, the script simply binds the module.

See the IF and EQS command descriptions in the *DOMAIN System Command Reference Manual* for further information.

In-Line Data

To include utility commands in a Shell script, use the special input redirection notation <<. For example, EDIT normally takes its commands from the standard input (usually the input pad). However, you can include EDIT commands in a script this way:

```
ED file <<!
editing requests
!
```

The lines between <<! and ! are called a **here document**. The Shell passes them to EDIT as if they were standard input. The character "!" is arbitrary; you can select any character to

begin and terminate the here document. The termination character you select must match the character that follows the << input redirection sequence. The here document's final line must only consist of the termination character. Place the termination character in the line's first character position. A here document may also contain the ^n argument substitution sequence.

Displaying Script Commands

To debug a Shell script, execute it with the SH (SHELL) command and include the -X option. This option writes each command line in the script to standard output immediately before execution; it provides each command's complete pathname, and fills in each argument. For example,

```
SH -x FOP <RET>
```

This command line executes the FOP script. Before FOP executes a command in the script, it writes the command line to the transcript pad.

The SH command also accepts a -V option. This option verifies script execution by writing command lines to standard output upon execution. (If you are typing input, the option echoes command lines as you type them.)

You can also control echoing by including the Shell commands VON, VOFF, XON, and XOFF in your script.

Shell Command Summary

Now that you're familiar with the Shell command environment, experiment with some of the commands in the following list. The summary does not include every Shell command, but it will give you a general idea of the types of services Shell commands provide.

Use the HELP system to determine the proper format, arguments, and options for each command. Read the *DOMAIN System Command Reference Manual* for detailed command descriptions.

FILE CONTROL COMMANDS

catenate__file	CATF	catenate files to standard output
copy__file	CPF	copy a file
copy__screen	CPSCR	copy screen to file for hardcopy output
delete__file	DLF	delete a file
move__file	MVF	move a file or directory
overstrike	OS	change backspaces in file to FORTRAN carriage control
print__file	PRF	print a file

TEXT CONTROL COMMANDS

change__pattern	CHPAT	change text in a file which matches a pattern
compare__file	CMF	compare two or more ASCII files
compare__sorted__file	CMSRF	compare two sorted ASCII files
delete__duplicate__lines	DLDUPL	delete duplicate lines within an ASCII file
file__length	FLEN	count words, lines, or characters in a file
find__pattern	FPAT	find a pattern in an ASCII file
find__pattern__block	FPATB	find a pattern block in an ASCII file
find__spelling__errors	FSERR	find spelling errors in an ASCII file
format__text	FMT	format text
format__multi__column	FMC	format text into multiple columns
sort__file	SRF	sort an ASCII file

DIRECTORY CONTROL COMMANDS

compare__tree	CMT	compare two directories and their files
copy__tree	CPT	copy a tree
create__directory	CRD	create a directory
delete__tree	DLT	delete a tree
list__directory	LD	list a directory
move__file	MVF	move a file or directory
naming__directory	ND	set or show the naming directory
salvage__directory	SALD	salvage a damaged directory
working__directory	WD	set or show the working directory

LINK CONTROL COMMANDS

create__link	CRL	create a link
delete__link	DLL	delete a link

OBJECT CONTROL COMMANDS

change__name	CHN	change the name of an object
catalog__object	CTOB	catalog an object in the name space
list__locked__objects	LLKOB	list locked objects
lock__object	LKOB	lock an object
object__type	OBTY	set the type of an object
uncatalog__object	UCTOB	uncatalog an object from the name space
unlock__object	ULKOB	unlock an object

ACCESS CONTROL COMMANDS

acl	ACL	show or copy the ACL of one or more objects
default__acfs	DEFACL	show or set default ACLs
edit__acl	EDACL	edit the ACL of one or more objects
salvage__acfs	SALACL	salvage the acfs on a volume

PROGRAM DEVELOPMENT COMMANDS

compile__c	CC	compile C program
compile__fortran	FTN	compile FTN program
compile__pascal	PAS	compile PASCAL program
bind	BIND	bind an object module
debug	DEBUG	language level debugger
histogram__pc	HPC	generate histogram of program counter

PROCESS CONTROL COMMANDS

abort__severity	ABTSEV	show or set the Shell abort severity
command__search__rules	CSR	show or set Shell command search order
list__address__space	LAS	list the objects mapped into the process address space
shell	SH	execute a Shell process
signal__process	SIGP	send signal to a process
process__statistics	PST	show information about created processes

NETWORK CONTROL COMMANDS

catalog__node	CTNODE	catalog a node in network root directory
list__connected__nodes	LCNODE	list the nodes connected in the network
list__users	LUSR	list users logged on to the network
network__service	NETSVC	set or show network service
network__statistics	NETSTAT	show the network statistics
uncatalog__node	UCTNODE	uncatalog a node from the network root directory

VOLUME CONTROL COMMANDS

dismount__volume	DMTVOL	dismount a logical volume
initialize__volume	INVOL	initialize a logical volume (off-line only)
list__volume__free__space	LVOLFS	list free space on mounted logical volumes
mount__volume	MTVOL	mount a logical volume

MISCELLANEOUS COMMANDS

date	DATE	print the current date and time
desk__calculator	DCALC	provide a desk calculator
help	HELP	display information about software
touchpad__mode	TPM	set or show touchpad mode
read__backup	RBAK	index or restore a file on magnetic tape
write__backup	WBAK	copy a file to magnetic tape
read__write__magtape	RWMT	read or write a magnetic tape produced by or for another manufacturer's computer system

Chapter 6

Controlling Access to Files and Directories

You can protect your files and directories from unauthorized use with access control lists (ACLs). Every object in the network has an ACL. It lists the people who can use the object, and specifies how each person can use (access) the object. For example, an ACL can authorize some users to read a file, and permit a few users to edit it. When a user process requests access to a network object, the system checks the object's ACL. If the ACL grants the user the appropriate access, then the system lets the process use the object. You can display and change ACLs using Shell commands.

Using the Access Control Commands

To display an ACL, use the Shell command `ACL` with the `-L` option. The next example shows the format of the ACL command and the system response:

```
$ ACL -l filename <RET>
Acl for filename:
user.project.organization.node      pndrwx
```

An ACL can contain one or more entries. The sample ACL shown above consists of a single entry with two main parts. The combination of usernames on the left is called a subject identifier (SID). The letters on the right (pndrwx) represent a list of access rights. Access rights determine how the SID can use the object.

The system gathers SID information during the log in procedure. A SID has four fields. The first field contains a username. The second and third fields contain the user's project and organization names. The fourth field identifies the user's node. The system assigns the same SID to every process a user creates before he or she logs off, (unless the user changes the SID with the `LOGIN` Shell command). When a process requests access to a file or directory, the system compares the process's SID with the ACL of the desired object. If the object's ACL includes a SID that matches the process's SID, the system grants the process the type of access rights specified in the ACL.

The following ACL contains one entry. (In a moment, we'll add a second entry):

```
$ ACL -l prog3 <RET>
Acl for prog3:
jon.doc.r_d.%      pndwrwx
```

This SID describes the user `JON`, on the `DOC` project, in the `R_D` organization. The `%` symbol appears in the SID's node ID field. This symbol matches zero or more characters — in this case any node ID. This means `JON` can use `PROG3` from any network node. (If the SID specified a particular node ID, `JON` could only access the file from that node.)

The rights (pndwrx) listed in this ACL grant JON complete access to the file. Protect (p) rights enable JON to change the file's ACL. The set of rights also includes node (n) rights. Node rights let JON change the SID's node ID specification. Other rights enable JON to delete (d), write (w), read (r) or execute (x) the file. (Tables 6-1 and 6-2 explain the types of rights that you can assign.)

To add SIDs and rights to an object's ACL, use the EDACL (EDIT__ACL) command with the -a (add) option. For example,

```
$ EDACL -a %%.r_d r prog3 <RET>
```

lets any user with the organization name R_D read Prog3. When you specify SID entries, you can omit trailing % symbols. In this example, we omitted the % symbol from the last field (the node ID specification). Thus, users in R_D can access the file from any network node.

Now, the ACL for PROG3 contains two entries:

```
$ ACL -l prog3 <RET>
Acl for prog3:
  jon.doc.r_d.%          pndwrx
  %%.r_d.%              -----r-
```

The system automatically places specific ACL entries (such as jon.doc.r_d.% before general ACL entries (such as %%.r_d.%). ACL entry order is significant because the system stops reading ACL entries as soon as it finds a SID that matches the process SID. If we modify the rights in the previous example to be the following,

```
jon.doc.r_d.%          -----
%.r_d.%                pndwrx
```

Then the system denies JON access, even though he is a member of the R_D organization.

The system ensures that *someone* always retains the right to change an object's ACL (protect right — p). If you change an ACL, and expect to change it again, *retain protect rights*. Otherwise, you forfeit the right to change the ACL.

You may receive the following error message when you change an object's ACL:

```
Prog3:
warning: object may not be readable by backup procedure
```

This means the ACL does not grant read access to users with the project name BACKUP. This could interfere with the system-wide magnetic tape backup procedure. Find out how your System Administrator performs backups. If necessary, use EDACL to add read rights for project BACKUP.

Directories have different kinds of access rights because you use directories in a different way from files. The following command displays the ACL for the directory //ET:

```
$ACL -l //et <RET>
Acl for //et:
  you.proj.org.%          pndcalr
```

Protect (p), node (n) and delete (d) have the same meaning for files and directories. Directory change (c) rights let you change directory entry names and delete links from the directory. Add (a) rights permit you to add files and subdirectories. Link (l) rights let you add links. Directory read (r) rights allow you to list directory entries.

To delete a directory you must have delete rights to its files and subdirectories. If the directory contains links, you must also have directory change (c) rights (which grant you the right to delete the links).

Table 6-1 summarizes the access rights you can assign to files and directories. You can specify rights individually or use abbreviations for certain kinds of rights. For example,

```
$ EDACL -a jon -owner memo <RET>
```

grants JON complete rights (pndwrx) to the file memo. Table 6-2 defines the abbreviations that specify commonly assigned rights.

To copy an object's ACL to another object, use the ACL command. For example,

```
$ ACL target_object source_object <RET>
```

assigns source_object's ACL to target_object. To merge duplicate ACLs, and delete unused ones, use the Shell command SALACL (SALVAGE_ACCESS_CONTROL_LIST). See the *DOMAIN System Command Reference Manual* for details.

Initial Access Control Lists (ACLs)

In addition to its own ACL, each directory has two initial ACLs — one for its new files and one for its new subdirectories. When you create a file or directory, the system assigns it the parent directory's initial ACL for that object type. To display a directory's initial ACL, use the ACL command with its -I option. For example,

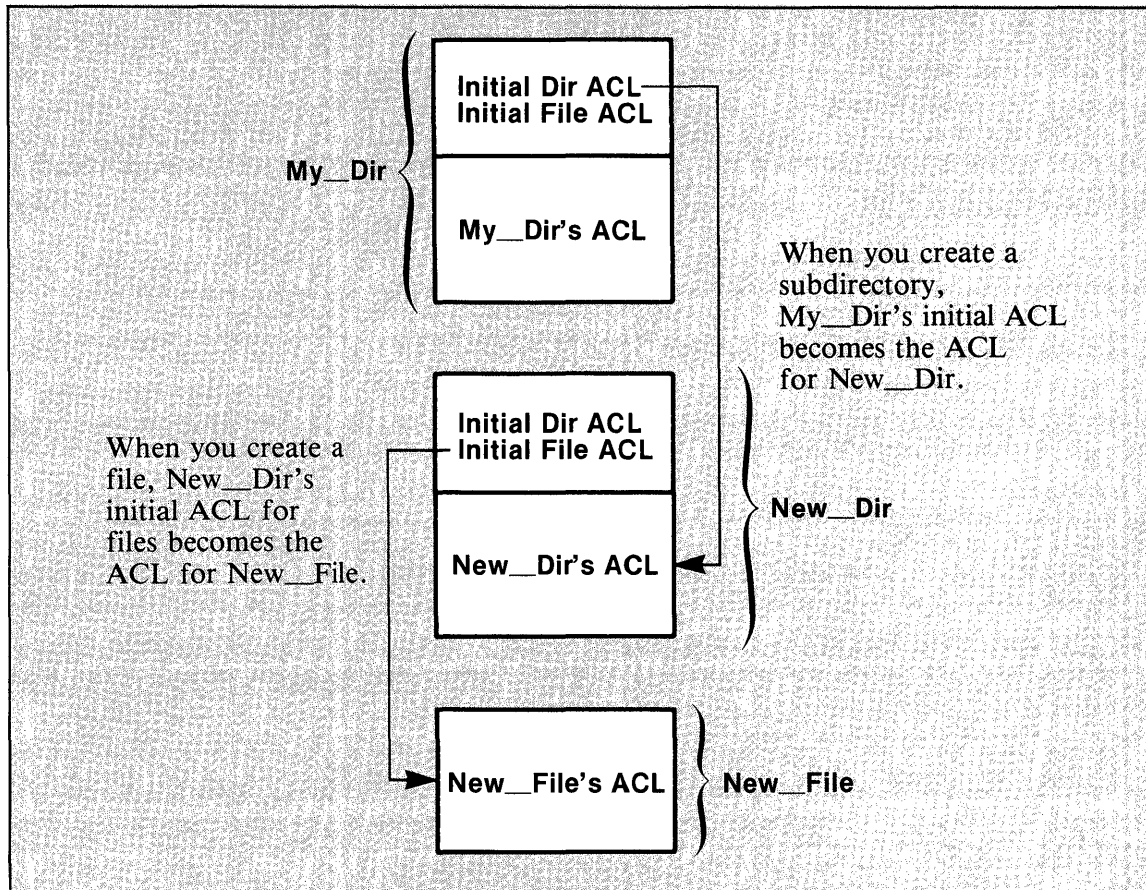
```
$ ACL -i //My_Node/My_Dir <RET>
Initial (default) acl for directories created under //My_Node/My_Dir
  %.%r_d.%                pndcalr
Initial (default) acl for files created under //My_Node/My_Dir
  %.%r_d.%                pndwrx
```

If you create a subdirectory in //My_Node/My_Dir, the system assigns the new directory My_Dir's initial ACL for subdirectories. If you create a file in //My_Node/My_Dir, the system assigns the new file My_Dir's initial ACL for files. (See Figure 6-1.)

When you create a new subdirectory, the subdirectory's initial ACLs match its parent directory's initial ACL. To change initial ACLs, use the EDACL or ACL command.

The system also assigns the appropriate initial ACL if you copy files or directories using the Shell commands CPF (COPY_FILE) or CPT (COPY_TREE). When you copy an object into a directory, the system assigns the destination directory's initial ACL to the newly-created object. Include the copy command option -SACL if you want the copied object to keep its original ACL. See the *DOMAIN System Command Reference Manual* for details about these commands.

Figure 6-1: Initial Access Control Lists



Creating Protected Subsystems

Usually, a program can access a file if the process running the program has the proper SID. At times, you may want to grant a program access rights to certain data files, regardless of the program's process SID. The term *protected subsystem* refers to a set of data files and a program that has special access rights to the files. The *DOMAIN System Administrator's Guide* explains how to set protected subsystem status in a file's ACL.

Table 6-1: Access Rights for Files and Directories

Access Right	Abbreviation	Capabilities Granted for:	
		Directories	Files
Protect	P	Change the object's ACL	
Node	N	Change the nodes from which users may access the object	
Delete	D	Delete the directory	Delete the file
Read	R	List entries	Read file contents
Write	W		Write to the file
Execute	X		Execute object file
Change	C	Change names and delete entries	
Links	L	Add links	
Add	A	Add files and subdirectories	

Table 6-2: Abbreviations for Commonly Assigned Rights

Term	Meaning	Directories	Files
-OWNER	All rights	PNDCALR	PNDWRX
-USER	All rights except ability to change ACL	DCALR	DWRX
-READ	File read access	not allowed	R
-EXEC	File read access Execute access to object files	not allowed	RX
-LDIR	List directories	R	not allowed
-ADIR	List directories and add entries	ALR	not allowed
-NONE	Grants no rights (Use to <i>deny</i> access)	none	none

Appendix A

Starting the Node

Normally, you won't need to start your node, it will already be running and you'll just log in. We provide this information to save you time in the event that someone shuts your node off (to perform maintenance or to connect a peripheral device) and then fails to turn it back on.

Nodes operate in two modes: normal and service. Before you execute the steps described below, make sure the NORMAL/SERVICE switch is set to NORMAL. (On the DN300 node, the NORMAL/SERVICE switch is on the back of the CPU card cage. On other nodes, the NORMAL/SERVICE switch is inside the CPU cabinet on the right-hand panel and/or on a printed circuit board. If the switches appear on both the panel and the circuit board, use the switch on the circuit board.)

Now, turn the power on. (The ON/OFF switch is on the back of DN300 nodes, near the electrical cord. On other nodes, a key or a button on the front of the CPU cabinet supplies power.)

The node executes a set of diagnostics (tests). If your node is diskless, a message gives its partner node's ID number. The AEGIS operating system starts and displays the message "Loading Display Manager". Our logo also appears on the screen.

When the prompt "Please log in:" appears, log in. Chapter 1 explains how to log in.

Appendix B

Initial Directory and File Structure

The following illustrations show how the system organizes the software that we supply with your node. Figure B-1 depicts the contents of the node entry directory (/). Figure B-2 shows the files and directories in the system software directory (/SYS). Figure B-3 contains the files and directories in the Display Manager directory (/SYS/DM). Finally, Figure B-4 illustrates the network management directory (/SYS/NET).

Figure B-1: The Node Entry Directory (/) and Subdirectories

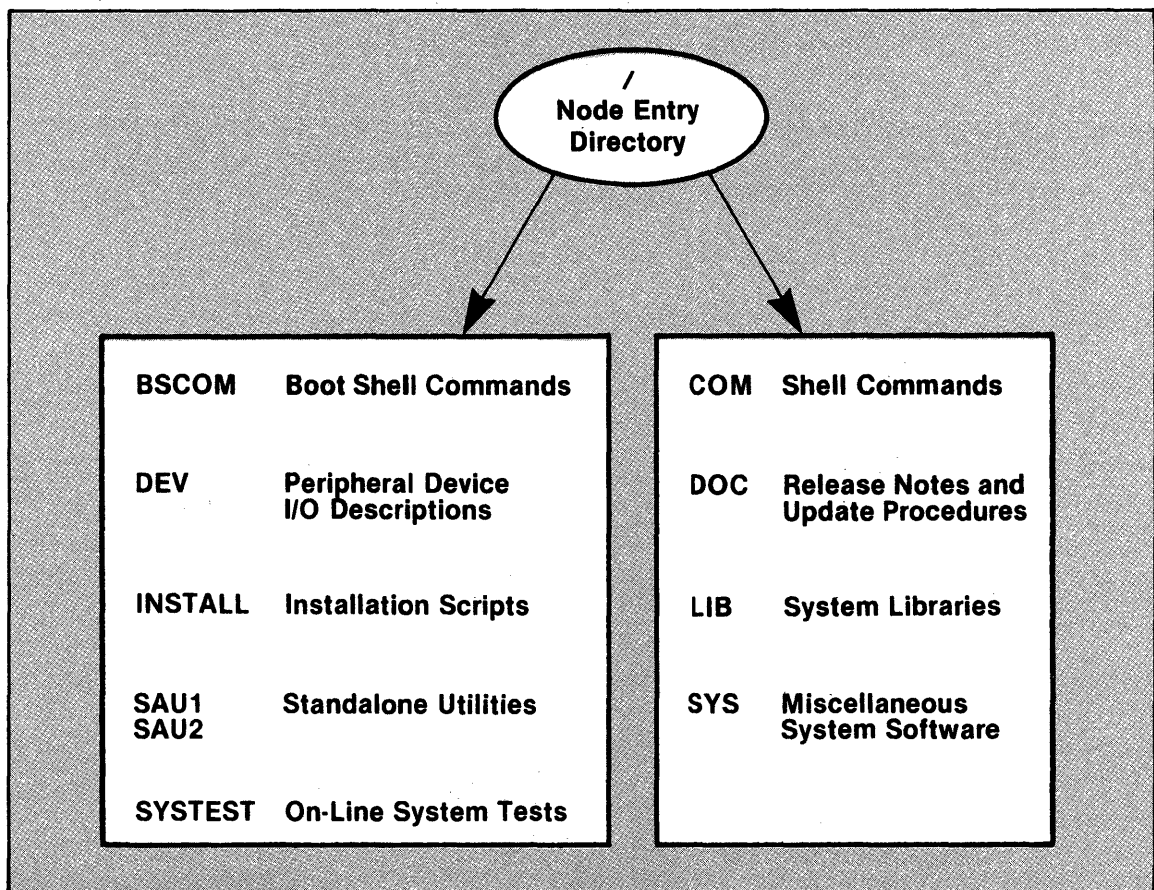


Figure B-2: The System Software Directory (/SYS)

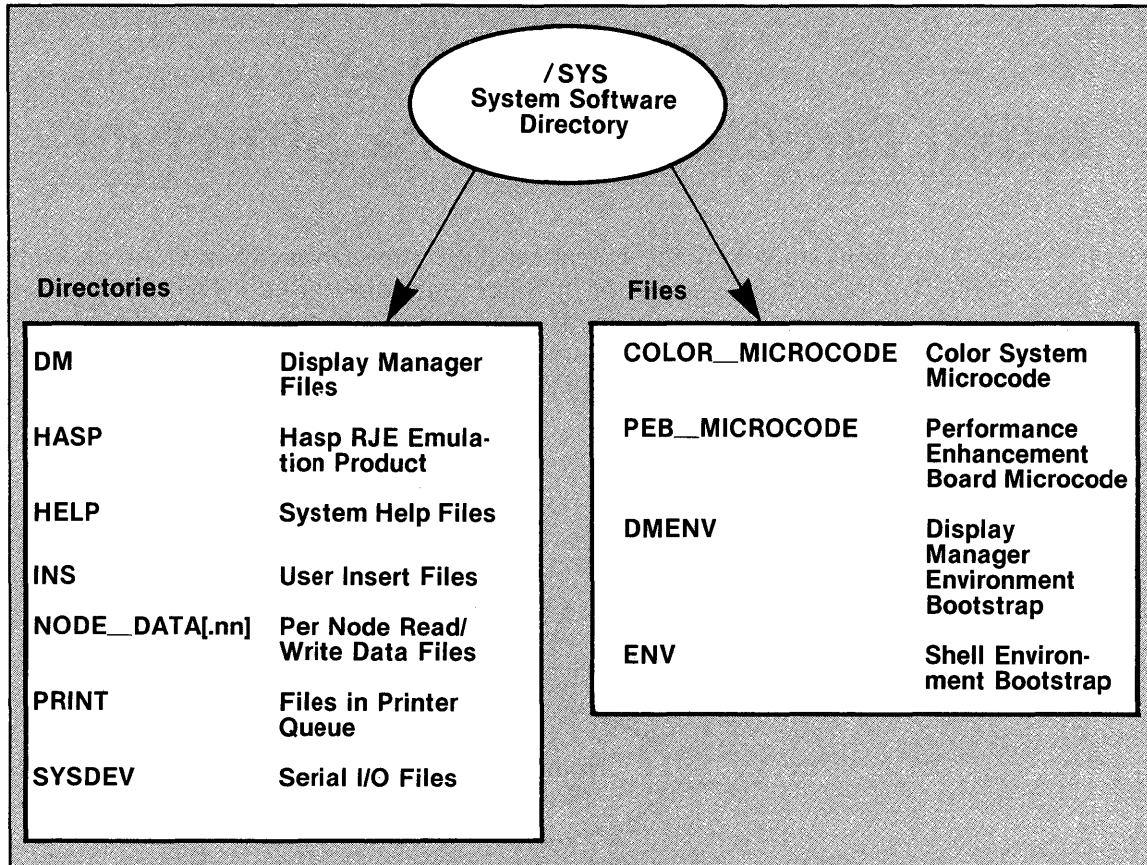


Figure B-3: The Display Manager Directory (/SYS/DM)

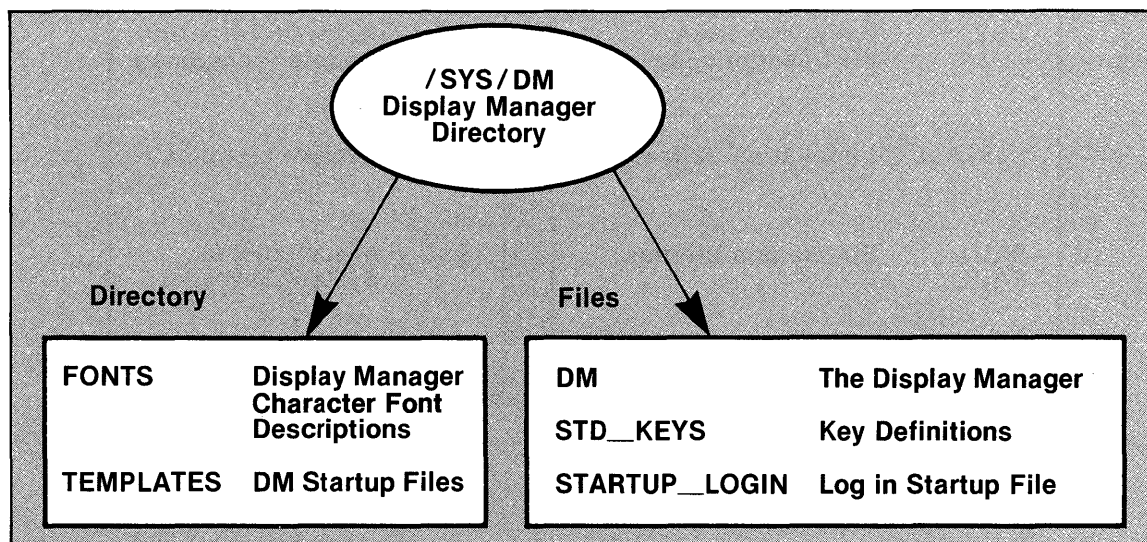
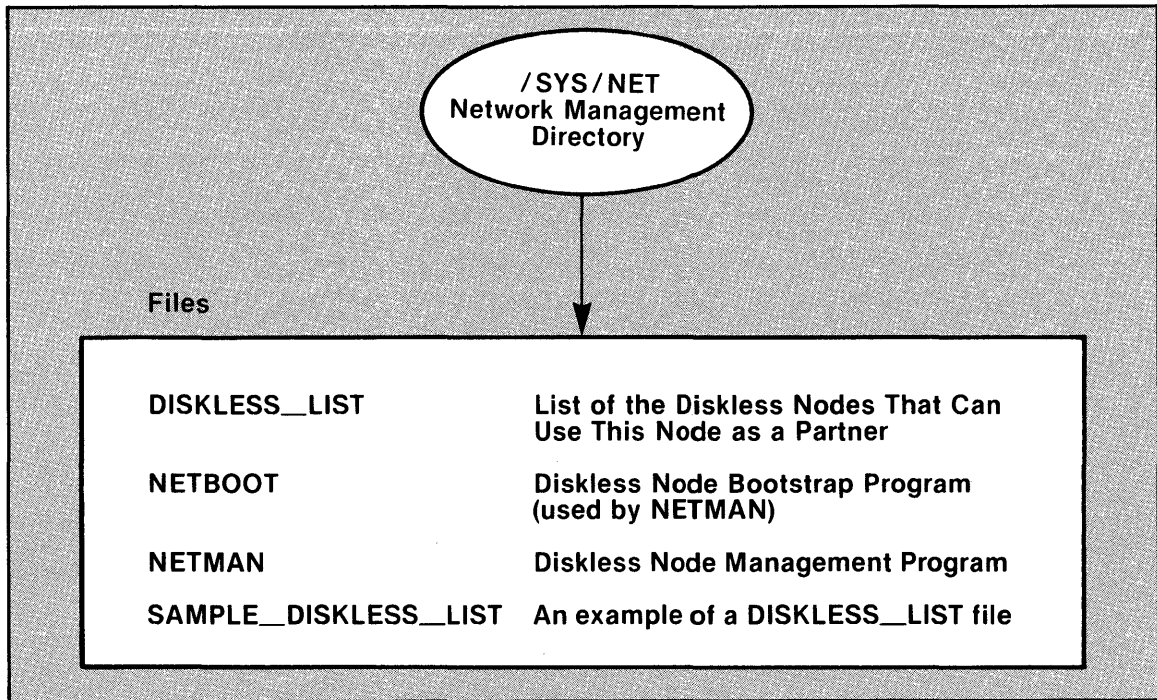


Figure B-4: The Network Management Directory (/SYS/NET)



Appendix C

Standard Suffixes

Use two-part names to indicate the kind of information that is in a file. For example, the name Test.FTN designates a file containing FORTRAN source code. A name ending in .BIN designates a file containing executable object modules. Use the set of suffixes listed in Table C-1 whenever possible. Some programs require that the input filenames end in a particular suffix. For example, the files you plan to input to the FORTRAN compiler must end in .FTN.

Table C-1: Standard Name Suffixes

Suffix	Meaning
.BAK	Backup file, written by Display Manager
.BIN	Binary file, produced by compiler
.DATA	Data file
.FTN	FORTRAN source code
.INS	Insert file
.LST	Listing file, produced by compiler
.MAP	Map file, produced by binder
.PAS	Pascal source code
.C	C source code

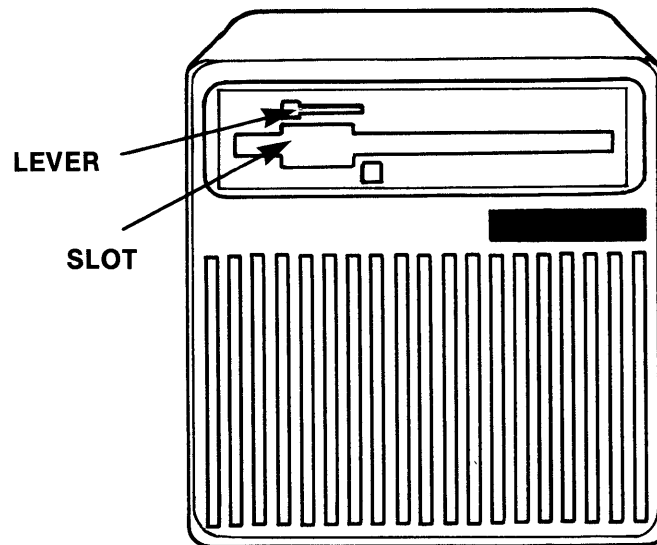
Appendix D

Using a Floppy Disk

Floppy disks provide an easy way to transfer information between nodes that do not belong to the same network. You can read or write files to or from a floppy disk. You may choose to store files you seldom use on floppies, and make more disk space available on your system. You can also save back-up copies of important files on floppy disks.

To use a floppy disk, you must be working on a node that contains a floppy disk drive. The upper portion of the disk storage option on a DN300 node contains the drive (see Figure D-1). The disk storage option is a small, separate box that holds a Winchester disk and a floppy disk drive. On other nodes, the front cabinet contains the drive (See Figure D-2).

Figure D-1: A Disk Storage Option Floppy Disk Drive



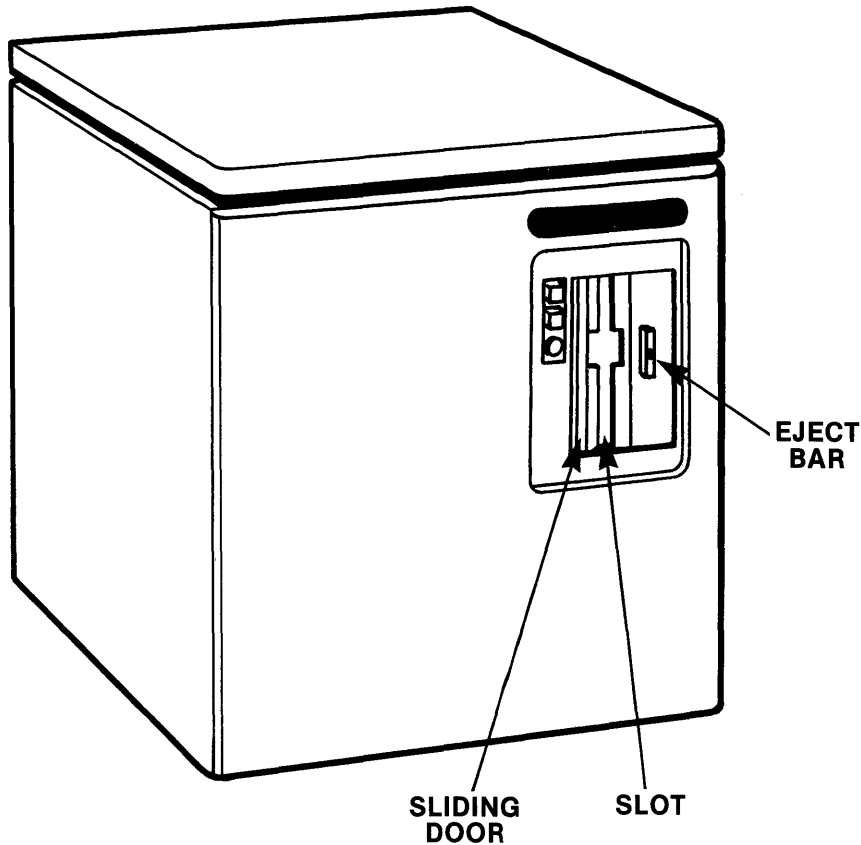
Looking at the front of a floppy drive, you see a slot and short eject bar (or lever). On nodes with the floppy drive on the cabinet, a long bar appears beside the slot. This bar is attached to a sliding door.

Using the floppy disk requires five steps:

1. Insert
2. Initialize
3. Mount
4. Dismount
5. Remove

This appendix explains how to perform each step.

Figure D-2: A Cabinet-Mounted Floppy Disk Drive



Inserting a Floppy Disk

To insert a floppy disk, follow these steps:

1. Take the floppy disk out of the paper envelope. *Do not remove the plastic jacket or you will damage the surface of the disk. Do not touch the surface of the disk, which is visible through cut-outs in the jacket.*
2. Look for a small slit in the side of the jacket. The slit prevents you from storing (writing) information on the floppy. If you want to write on the floppy, cover the slit with a glue-backed tab. Tabs are usually included in the box that holds new floppy disks. Fold the tab over the slit so that *the tab completely covers the slit on both sides of the jacket.*
3. Leave the floppy in its plastic jacket. If you are working on a node with the floppy drive on the front of the cabinet, insert the floppy in the slot, with the slit or tab pointing into the slot as shown in Figure D-3. Push the floppy in (with the label facing left) until it “clicks” into place. Grasp the long bar and slide the door over the slot firmly, until it too “clicks” into place. If you are using a DN300 node insert the floppy as shown in Figure D-4, turn the lever down and close the door.

Figure D-3: Loading a Floppy Disk in a Cabinet-Mounted Floppy Drive

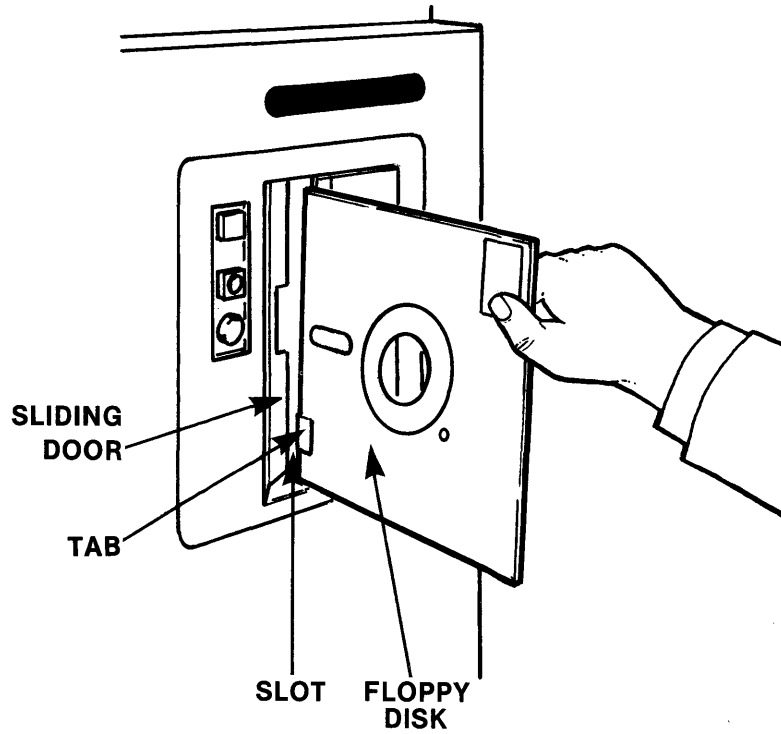
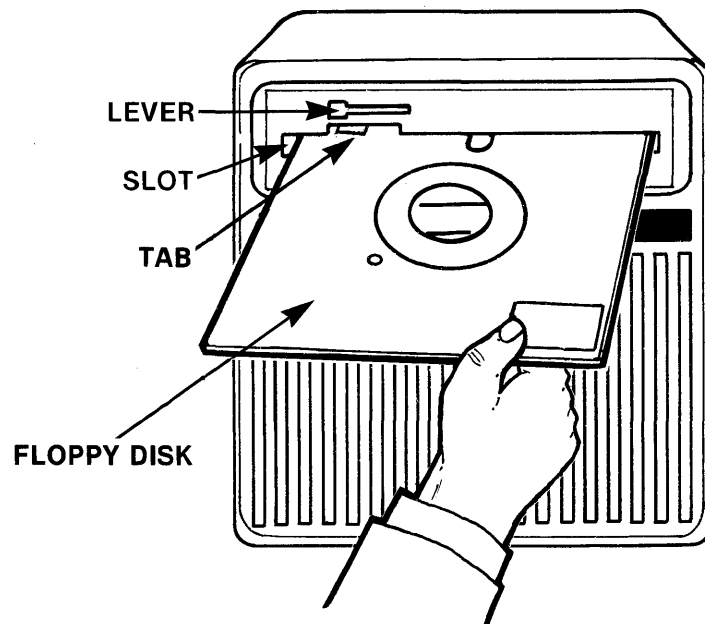


Figure D-4: Loading a Floppy Disk in a Disk Option



Initializing a Floppy Disk

You must initialize a floppy before you can store files on it. If you're using a new floppy, it has probably never been initialized. (If you are not sure if the floppy has been initialized, try to mount it as described in the next section.)

It takes a few minutes to initialize a floppy. To do so, prepare the floppy for writing by applying the sticky, write-enable tab, and insert it in the disk drive. Log in and type

```
$ INVOL <RET>
```

This executes the Shell program INITIALIZE__VOLUME. The program prompts you to supply some information. Respond as shown in Figure D-5.

Figure D-5: Initializing A Volume

```
$ INVOL <RET>

invol (initialize_volume), revision x.x date

Options are:
 1 - initialize virgin physical volume.
 2 - initialize partial physical volume, preserving existing logical volumes.
 3 - re-initialize an existing logical volume.
 4 - delete a logical volume.
 5 - list logical volumes.
 6 - list badspots on volume.
 7 - input and record badspot information.
 8 - create or modify an os paging file on an existing logical volume.
 9 - add to existing badspot list.

Option: 1 <RET>

Disk types are:
  W - Winchester
  S - Storage module
  F - 2-sided double density floppy

Disk type: f <RET>

Verification options are:
 1 - no verification
 2 - write all blocks on the volume
 3 - write and re-read all blocks on the volume

Enter verification option: 3 <RET>

Expected average file size, in blocks (CR for default, 5 blocks): <RET>

For each logical volume to be formatted, enter the logical volume
size, followed by the name, in the form "size, name". Up to 10
volumes may be specified. Terminate input with a blank line.

There are 1231 blocks available.

volume 1: 1231 <RET>

Enter badspots between physical disk addresses 1 and 4CF, one per line.
Badspots must be input in (hex) physical disk address form.
Terminate badspot entry with a blank line.

Is the badspot information you entered correct? y <RET>
.
.
.

Initialization complete.

Anything more to do? n <RET>
$
```


After you supply the necessary information, INVOL prepares the disk for file storage. You will hear some clicking as the disk drive operates. When this process is complete, INVOL asks if there is anything more to do. Answer no and proceed to the next section.

Mounting a Floppy Disk

Mounting the floppy disk, or mounting the logical volume, makes the storage areas on the floppy accessible to the operating system. Follow the steps below to mount the floppy.

1. Load the floppy and log in if you have not already logged in.
2. Initialize the floppy if it has not been initialized. (See the initializing procedure described above.)
3. Type MTVOL (MOUNT__VOLUME) and command arguments as follows:

```
$ MTVOL F flpname <RET>
```

The first argument, “F” (Floppy), specifies the disk type. The argument, “flpname”, stands for the volume entry directory name. This name will become the name of the top-level directory for the files on the floppy. Make a note of the name you choose, you’ll need to use it when you dismount the floppy.

4. If the floppy you are using has not been initialized, you will receive the following message when you try to mount it:

```
Unable to mount volume, bad disk format
```

Refer to the previous section for information on how to initialize a floppy.

5. Set your working directory to the volume entry directory. If you just initialized the floppy, it will contain only the “sysboot” file created by the initialization program. You can create directories and files on the floppy, list directories, copy to and from the floppy, and read or edit on the floppy just as you normally do.

Dismounting and Removing a Floppy Disk

You must *dismount the floppy before you remove it from the drive, or you may lose the information stored on it.* Dismounting a floppy makes the objects on it inaccessible to the operating system. You can copy these objects to your node before you dismount the floppy using the Shell commands CPT (COPY__TREE) and CPF (COPY__FILE).

To dismount and remove a floppy disk, follow these steps:

1. Set your working directory to a directory that is not on the floppy.
2. Type DMTVOL (DISMOUNT__VOLUME) and command arguments in a Shell input window as follows:

```
$ DMTVOL F FLPNAME <RET>
```

“FLPNAME” stands for the volume entry directory (you specified this name when you mounted the floppy).

3. To remove a floppy from the disk drive, press the eject bar next to the slot, or, if you are using a DN300 node, turn up the bar above the slot. The floppy will pop out of the slot. Replace the floppy in its envelope, placing the slit or tab at the bottom of the envelope. Place a label on the floppy jacket that lists the files and directories on the floppy.

Glossary

Access Control List (ACL)	You can protect your files and directories from unauthorized use with access control lists (ACLs). Every object in the network has an ACL. It lists the people who can use the object, and specifies how each person can use (access) the object.
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 that is hidden by an overlapping window.
Argument	See command argument.
Command	An instruction you give a program.
Command Argument	An object name you include in a command line to specify the object upon which the command will act. You must use one or more spaces to separate commands from arguments, and to separate arguments from each other. To include blank spaces within an argument, supply the argument in single or double quotation marks.
Command Option	Information you provide on a command line to indicate the type of action you want the command to take. Precede each option with a hyphen. Do not include any space between the hyphen and the option. (Also see “default.”)
Control Character Sequence	A keystroke combination that consists of <CTRL> followed by an alphabetic character key. To issue a control character sequence, you must hold <CTRL> down while you press the appropriate character key. DOMAIN control character sequences issue certain DM commands. For example, the control character sequence CTRL/C issues the DM copy command XC. To issue a control character sequence, such as CTRL/C, type the letter C while you press <CTRL>.
Cursor	The small, blinking box initially displayed in the screen’s lower, left corner.
Data	Information processed by a computer.
Default	Most programs allow you a choice of one or more options. If you do not specify an option, the program automatically assigns one. This automatic option is called the default.
Disk	A thin, record-shaped magnetic material that stores data as patterns of magnetization. The system uses <i>heads</i> (similar to heads in tape recorders) to read and write data on concentric disk tracks. The disk spins rapidly, and the heads can read or write data on any disk track during one disk revolution. This is why disks are called <i>random access media</i> .

Diskless Node	A node that has no disk, and therefore uses the disk of another node.
Directory	A special type of object that contains information about the objects beneath it in the naming tree.
Display Manager (DM)	The program that executes commands that start and stop processes, and commands that open, close, move or modify windows.
DM Function (DMF) Keys	Single keys which invoke DM commands.
DOMAIN	Distributed Operating Multi-Access Interactive Network.
DOMAIN System	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. Ideally, each system user has his or her own node.
File	The basic named unit of data stored on disk. A file can contain a memo, manual, program or picture.
Filters	Commands that copy standard input to standard output and make some changes along the way. (See Chapter 5 for examples).
Function Keys	See Display Manager Function Keys.
Here Document	Specially-marked lines in a Shell script that a utility invoked in the script uses as standard input. (See Chapter 5 for examples.)
Home Directory	Your initial working directory. Your user account specifies the name of your home directory.
Initial ACLs	In addition to its own ACL, each directory has two initial ACLs — one for its new files and one for its new subdirectories. When you create a file or directory, the system assigns it the parent directory's initial ACL for that object type. (See Chapter 6 for an example.)
Initial Working Directory	The working directory of the first user process created after you log in.
Input Window	The window that displays a program's command prompt, and any instructions you type.
Insert Mode	Insert mode enables you to change text displayed in windows. You can modify text by repositioning the cursor and inserting characters. The rest of the line moves right as you insert additional characters. Use <INS> to exit and enter insert mode. (On some keyboards this key is labeled INS MODE.)

Link	A special type of object that points from one place in the naming tree to another. A link contains the name of another object.
Logging In	To begin using the system, log in by typing the L command followed by your username(s) and password. This creates your initial user process.
Logging Off	To stop using the system, log off by typing the DM command LO. This prevents others from using your account.
Main Memory	The node's primary storage area. It stores the instruction that the node is executing as well as the data it is manipulating.
Memory	Any device that can store information.
Name	A 1 to 32 character string associated with a file, directory or link. A name can include alphanumeric characters, periods (.), underscores (_), or dollar signs (\$). All names must begin with a letter or a dollar sign.
Naming Directory	In addition to its working directory, each process uses a naming directory. Like the working directory, the naming directory points to a certain destination directory. You can display or set the destination directory at any time using the Shell command ND (NAMING_DIRECTORY). The system uses your home directory as the initial naming directory. After you set the naming directory, you can use the shorthand symbol "~" in place of the naming directory name.
Naming Tree	A hierarchial tree structure that organizes every network object.
Network	Two or more nodes sharing information.
Network Object	See object.
Network Root Directory	The top directory in the naming tree, represented by a pair of slashes (/). Each node contains a copy of the network root directory.
Node	A network computer. Each node in the DOMAIN system 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.
Node Entry Directory	A subdirectory of the network root directory. The node entry directory, represented by a single slash (/) is the top directory on each node. Diskless nodes share the node entry directory of their disked partner node.
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's response to your command.
Pad	A temporary, unnamed file that holds the information you display in a window. A window can display an entire pad, or only show part of the pad.
Parent Directory	The directory one level above the current working directory. A backslash (\) represents the parent directory.
Partner Node	A node that shares its disk with a diskless node.
Password	The final name you enter when you log in. The system does not display your password in the process input window.
Pathname	A series of names separated by slashes that describe the path the operating system must take to get from some starting point in the naming tree to a destination object. It begins with the starting point's name, and includes every directory name between the starting point and the destination object. A pathname ends with the destination object's name. A pathname may not exceed 256 characters, including the slashes.
Pipe	If you place two commands on one line, and separate them with a vertical bar, the Shell uses the output of the first command as input to the second command. The vertical bar () between the commands is called a pipe.
Pipeline	A command line that uses pipes and filters.
Process	An executing program.
Prompt	A message or symbol displayed by a program to indicate that it is ready for your input. The Shell prompt is a dollar sign (\$) and the DM prompt is "Command:".
Rights	An access control list contains a set of access rights. These rights determine if and how a process can use an object.
Root Directory	See Network Root Directory.
Script	A file that you create which contains one or more Shell command names.
Shell	A command line interpreter program used to invoke operating system utility programs.
Shell Commands	An instruction you give the system to execute a utility program. For example, when you type DATE the Shell invokes the utility program that displays the date.
Software	Programs.

Subject Identifier (SID)	The system assigns the same SID to every process a user creates before he or she logs off. The system gathers SID information during the log in procedure. A SID has four fields. The first field contains a username. The second and third fields contain the user's project and organization names. The fourth field identifies the user's node. When a process requests access to a file or directory, the system compares the process's SID with the ACL of the desired object. If the object's ACL includes a SID that matches the process's SID, the system grants the process the type of access rights specified in the ACL.
System Administrator	The person responsible for system maintenance and security at your installation.
Transcript Pad	The process output window provides a view to its transcript pad. The transcript pad contains a record of your interaction with the process.
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 (username), and his or her password. User accounts also contain project and organization names. The system uses this information to determine who can use the system, and what resources they can use.
Username	The first name you type to log in.
Utilities	Programs provided with the operating system to perform frequently required tasks.
Window	Display management software lets you create several different views, or windows on the screen. Windows provide openings on the screen through which you view information stored in the system. 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.
Window Legend	The area of a window that displays window status information.
Working Directory	The default directory in which a process creates or searches for objects.

Index

Primary page references are listed first. The letter “f” means “and the following page”; “ff” means “and the following pages”.

@sign 5-3

A

ABORT key 4-4
ABTSEV (abort__severity) 5-12
access
 control commands 5-12, 6-1ff
 control lists (ACLs) 6-1ff
 rights 6-1ff
 to files and directories 6-1ff
ACL (access-control-list) 5-12
AGAIN key 4-6
alarm
 DM window 1-6
 responding to 1-11f
arguments 5-1
 substituting 5-7ff
arrow keys 1-6f, 4-3, 4-22
asterisk (*) symbol 5-5f

B

BAK suffix 4-4, 4-13
BACK SPACE key 1-8f, 3-3, 4-8, 4-10
backslash character (/) 2-3ff
BINARY option 3-6
BIND (bind) 5-12
binding 3-6f

C

C program development 3-7
CC (compile__c) 5-12
case
 comparison search 4-12, 4-8
 sensitivity 4-13
CATF (catenate__file) 5-10
CHAR DEL key 1-9, 3-3, 4-8, 4-16
CHN (change__name) 5-11
CHPAT (change__pattern) 5-10
CMD key 1-7, 4-2f
CMF (compare__file) 5-10
CMSRF (compare__sorted__file) 5-10
CMT (compare__tree) 5-11
COM directory 2-6, 5-1
commands 1-1
 access control
 arguments to 5-1, 5-3
conditional statements 5-9

cut and paste 3-5
directory control 5-10
Display Manager 1-4, 1-9ff
file control 5-11
format for 5-3
link control 5-11
names for 5-2
network control 5-12
object control 5-11
options for 2-2, 5-3
process control 5-12
program control 5-12
search and substitute 3-5
search rules for 5-1
Shell 1-4, 1-8, 5-10ff
text control 5-10
undoing previous 4-13
volume control 5-12
command line processing 5-1
Command prompt 1-4f
control character sequences 1-9f
COPY key 3-5, 4-8, 4-11
copying text 4-6, 4-11
correcting errors 1-9, 3-3
CPF (copy__file) 5-10
CPSR (copy__screen) 5-10
CPT (compare__tree) 5-11
CRD (create__directory) 5-11
CRF (create__file) 4-3
CRL (create__link) 2-6, 5-11
CRUCR (create__user__change__request) iv
CSR (command__search__rules) 5-12
CTNODE (catalog__node) 5-12
CTOB (catalog__object) 5-11
CTRL/B 1-10, 4-6
CTRL/C 3-5, 4-8, 4-11
CTRL/E 3-5, 4-8, 4-11
CTRL/G 1-11, 4-4f
CTRL/K 4-14
CTRL/L 4-14
CTRL/M 3-1, 4-8
CTRL/N 1-12, 4-3, 4-4, 4-14
CTRL/O 3-5, 4-8, 4-11
CTRL/P 1-9, 4-4f
CTRL/Q 4-14
CTRL/R 4-8, 4-12
CTRL/T 1-10, 4-6
CTRL/ < TAB > 4-15
CTRL/U 4-8, 4-12
CTRL/W 4-4
CTRL/X 4-8, 4-12
CTRL/Y 3-5f, 4-3f

CTRL/Z 1-12, 4-8, 4-16, 4-21
cursor 1-4
 control keys 1-6f, 4-2ff
 moving 1-6ff
CUT key 3-5, 4-8, 4-11
cut and paste commands 3-5, 4-11

D

dash (-) symbol 5-6
DATE (date) 5-13
DCALC (desk__calculator) 5-13
DEBUG (debug) 5-12
DEFACL (default__acls) 5-11
deleting text 4-10
directional keys 1-7
directory 2-1ff
 COM 5-1, 2-6
 Display Manager (/SYS/DM) B-2
 home 2-3
 initial structure B-1
 initial working 2-2
 naming (~) 2-5f
 network management (/SYS/NET) B-3
 network root (//) 2-2, 2-1, 2-4ff
 node entry 2-2, 2-1, 2-4ff, B-1
 parent 2-3f, 2-6
 system software B-2
 working 2-2ff, 2-6
Disk Storage Option D-1ff
diskless node A-1, B-3
display
 environment, managing 4-17
 initial 1-4
 landscape 1-1, 1-3
 portrait 1-1f
Display Manager (DM) 1-1
 directory (/SYS/DM) B-1
 command summary 1-4, 1-9ff, 4-15ff
 using 4-1ff
DLDUPL (delete__duplicate__lines) 5-10
DLF (delete__file) command 5-10
DLL (delete__link) 5-11
DLT (delete__tree) 5-11
DM (see also Display Manager)
 alarm window 1-6
 commands 1-4f, 41ff
 function keys 1-9f, 1-12
 keys 1-7
DMTVOL (dismount__volume) 5-12
dollar sign 3-1
 prompt 1-4f
DOMAIN System 1-1

E

EDACL (edit__acl) 5-11, 6-2ff
EDIT key 3-1ff, 4-6
edit pad 4-6
edit window
 closing 3-5

 opening 3-3
editing
 files 3-3ff
 functions 4-7f
 a pad 4-7, 4-18
end-of-file (EOF) mark 4-10
EOL key 4-8, 4-10
EQS 5-9
EXIT key 4-5

F

F1 key 4-9, 4-8
F2 key 4-6
F3 key 4-6
F6 key 4-8
F7 key 4-8
F8 key 4-6, 4-4
filenames, entering 3-1ff
files 2-1, 3-1ff
 creating 3-1ff
 editing 3-3ff
 initial structure B-1
 opening window to 3-1ff
 reading 3-1ff
filters 5-7
FLEN (file__length) 5-10
floppy disk
 drive D-1ff
 initializing D-4f
 mounting and dismounting D-5
 using D-1ff
FMC (format__multi__column) 5-10
FMT (format__text) 5-7, 5-10
FOP 5-7ff
FORTRAN program, developing 3-6f
FPAT (find__pattern) 5-10
FPATB (find__pattern__block) 5-10
FSERR (find__spelling__errors) 5-10
FTN (compile__fortran) 3-6, 5-12
function keys (see also DM function keys) 1-9f

G

GROW key 4-4f

H

HELP (help) 1-8, 5-13
here document 5-9
HOLD/GO key 4-4, 4-6
hold mode 4-6
home directory 2-3
HPC (histogram__pc) 5-12

I

IF 5-9
in-line data 5-9
input
 redirecting 5-4ff
 reading standard 5-6

insert mode (INS) 1-9f, 3-3, 4-8f,
/overstrike 4-9
inserting text 4-9
INS (insert__mode) 1-9f, 3-3, 4-9
INVOL (initialize__volume) 5-12

K

KD 4-1
keyboard
 DN300 1-10
 DN4xx and 6xx 1-10
keys
 cursor control 1-7f
 DM 1-7
 DM function (DMF) 1-9f
 pad control 4-10ff
 pad editing 4-13ff
 predefined 4-1ff
 Shell process creation 4-13f
 Shell process termination 4-14
 window control 4-4ff
 window and pad creation 4-3ff

L

landscape display 1-1, 1-3
LAS (list__address__space) 5-12
LCNODE (list__connected__nodes) 5-12
LD (list__directory) 2-2ff, 5-11
LINE DEL key 1-9, 3-3, 4-8, 4-16
link 2-2, 2-6
LKOB (lock__object) 5-11
LLKOB (list__locked__objects) 5-11
LOGIN 6-1
logging in 1-1ff
logging off 1-12
LUSR (list__users) 5-12
LVOLFS (list__volume__free__space) 5-12

M

MARK key 1-11
mouse 1-8
 changing window size 1-11
 reading files 3-3
MOVE/GROW key 1-11, 4-4f
MTVOL (mount__volume) 5-12
MVF (move__file) 5-10f

N

naming directory (~) 2-5f
naming tree 2-1
ND (naming__directory) 2-5, 5-11
network
 DOMAIN 1-1
 management directory (/SYS/NET) B-3
NETSTAT (network__statistics) 5-12
NETSVC (network__services) 5-12
network root directory (//) 2-2, 2-1, 2-4, 2-6
NEWLINE character 4-8

NEXT WNDW key 1-7, 4-2f
node entry directory 2-2, 2-1, 2-4ff, B-1
node 1-1
 ID number A-1
 starting A-1
NORMAL/SERVICE switch A-1

O

objects 2-2f
OBTY (object__type) 5-11
output, redirecting 5-4
OS (overstrike) 5-7, 5-10

P

pad 1-10ff
 control keys 4-5ff
 creating windows to 4-16
 edit 4-1ff, 4-3f, 4-16
 input 4-1f
 moving under a window 4-6, 4-16
 moving to top and bottom 4-6
 read 4-3f
 transcript 1-10ff, 4-1f
parent directory 2-3
PAS (compile__pascal) 5-12
Pascal program development 3-7
password 1-2ff
 changing 1-4
paste buffers 4-7
PASTE key 3-5, 4-8, 4-11
pathname 2-3ff
period (.) 3-1
pipeline 5-6
pipes (:) and filters 5-7
PN 1-12
points and regions, defining 4-3
POP key 1-11f, 4-4f
portrait display 1-1f
PP (pad__page) 4-7
PRF (print__file) 3-7f, 5-7, 5-10
Print Server process 3-7
printing a file 3-7
process 1-1
 controlling 4-16
 creating 4-13, 4-15
 stopping 1-12, 4-14, 4-16
program
 developing 3-6f
 stopping 4-16
prompt 1-1
protected subsystem 6-4
PST (process__statistics) 5-12

R

range of text, defining 4-15
RBAK (read__backup) 5-13
READ key 3-1, 4-6

read pad 4-3f
read-only pad 4-4
read/write mode 4-9
reading files 3-1ff
redefining DM keys, 4-1
redirecting input/output 5-6ff
region 4-3
regular expression 4-8, 4-17
RETURN key 1-3, 4-8
RWMT (read__write__magtape) 5-13

S

SALACL (salvage__acls) 5-11
SALD (salvage__directory) 5-11
SAVE key 4-13
SC 4-12
scrolling 4-7f
search and substitute commands 3-5, 4-10ff
SH (shell) 5-9, 5-12
Shell 1-1
 using the 5-1ff
Shell commands 1-4
 entering 1-8
 summary 5-10ff
SHELL key 1-9, 4-20
Shell scripts
 debugging 5-9
 writing 5-7
SIGP (signal__process) 5-12
SO 4-8, 4-13
slash character (/) 2-4ff
SRF (sort__file) 5-10
string 4-11
SID (subject identifier) 6-1ff
substituting arguments 5-7
suffixes, standard C-1
System Administrator 1-2
system
 organization 2-1
 software directory (/SYS) B-1

T

TAB__SET 4-2, 4-15
TAB key 4-2
touchpad 1-6f
TPM (touch__pad__mode) 1-8, 5-13
transcript pad 1-10ff, 4-1

U

UCR iv
UCTNODE 5-12
UCTOB (uncatalog__object) 5-11
ULKOB (unlock__object) 5-11
underscore character(_) 3-1ff, 5-3
UNDO key 4-8, 4-13
undoing previous commands 4-13
updating an edit file 4-13
user account 1-2

user change request iv
username 1-2f
utilities 1-1

V

video display 1-1

W

WD (working__directory) 2-2ff, 5-11
WBAK (write__backup) 5-13
wildcards 5-6f
window legend 1-6
windows 1-1
 alarm 1-6
 control keys 4-4f
 creating 4-3ff
 changing size 1-11
 closing 4-5
 edit 3-3ff, 4-3f, 4-16
 input 1-5
 managing 4-4ff, 4-17
 moving 4-4ff
 output 1-6
 popping (pushing) 4-5
 position on landscape display 1-6
 position on portrait display 1-5
 read 3-1, 3-3, 4-3, 4-24
working directory 2-2f
WP (window pop) 1-9

READER'S RESPONSE

We use readers' comments to revise and improve our manuals.

Document Title: Getting Started With Your DOMAIN System

Date of Publication: October, 1983

What is the best feature of this manual? _____

Please list any errors, omissions, or problem areas in the manual. (Identify errors by page, section, figure, or table number wherever possible.)

What type of user are you?

- Systems programmer; language _____
 Applications programmer; language _____
 Student programmer
 User with little programming experience

How often do you use the DOMAIN system? _____

Nature of your work on the DOMAIN system: _____

Your name Date

Organization

Street Address

City State Zip/Country

No postage necessary if mailed in the U.S. Fold on dotted lines (see reverse side), tape, and mail.

cut or fold along dotted line

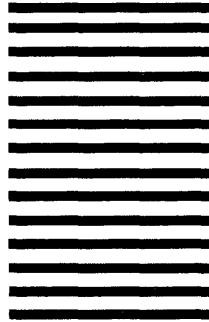
old



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 100 CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE



APOLLO COMPUTER, INC.
16 Elizabeth Drive
Chelmsford, MA 01824

Attn.: Software Documentation
Research and Development

old

apollo
computer inc.

16 Elizabeth Drive, Chelmsford, MA 01824