Educational Services

digital™

VMS Utilities and Commands I
Student Workbook – Volume I

EY-3501E-SA-0002

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem–10	PDP	VT
DECSYSTEM–20	PDT	
DECUS	RSTS	
DECwriter	RSX	

digital™

Second Edition, December 1988

This document was prepared using VAX DOCUMENT, Version 1.0

CONTENTS

FIGURES

TABLES

About This Course

INTRODUCTION

This *VMS Utilities and Commands* course is designed to show you how to perform typical nonprivileged operations on a VMS system by entering commands at a terminal.

This course is self–paced, which means that you can learn at whatever rate is comfortable. Note, however, that the course is not computer–based; there is no computer program that instructs you. Instead, your instruction is provided in this *Student Workbook.*

Instead of a teacher, you have a *course administrator* and a *subject matter expert.* In some cases, the same person functions as both.

The course administrator manages the course in general. The administrator makes sure that you have easy access to the system and online course materials. As you finish each module, the administrator records your progress.

The subject matter expert helps you if you have a technical question. Before you consult the expert, however, you are expected to read the course materials and references in an effort to answer the question yourself.

The text is divided into a number of units, or *modules*, each designed to cover a well–organized topic, or group of topics. Most modules are divided into smaller units, each with its own examples and exercises.

This *Student Workbook* describes the contents of the course, and suggests ways in which you can most effectively use its materials. This guide discusses the following topics:

- Course Description
- Course Prerequisites
- Course Goals
- Course Nongoals
- Equipment/Special Tools
- Course Resources
- Course Organization
- Course Map and Description
- Course Conventions

COURSE DESCRIPTION

This *VMS Utilities and Commands* course describes the working environment of a VMS system and introduces commonly required operations that you can perform by entering commands at an interactive terminal.

Among the major topics covered by the course are:

- Using online and printed VMS documentation to obtain help and other information
- Creating, editing, and maintaining text files
- Communicating with other users on a system and network
- Using logical names and symbols to tailor your working environment
- Writing and using command procedures
- Using disk and tape volumes, both public and private
- Submitting batch and print jobs

COURSE PREREQUISITES

There are no prerequisites for this course. However, students can derive the greatest benefit from this course by having:

- A basic knowledge of a computer system.
- The ability to work on a system using an interactive terminal.

COURSE GOALS

To effectively use the nonprivileged facilities of the VMS system, you should be able to perform the following operations by entering commands at an interactive terminal:

- Use online and printed documentation to obtain information about VMS features
- Use the DIGITAL Command Language (DCL) in a network environment
- Create and maintain collections of text, data, and program files
- Communicate with other users on your system and on a network
- Communicate with operators to request information and services
- Use logical names and symbols to modify your working environment
- Create and control command procedures, batch jobs, and print jobs
- Use private disk and tape volumes to back up and store personal files

COURSE NONGOALS

This course does NOT address the following:

- The MCR command language interpreter or other RSX utilities that reside on your system.
- The syntax of any programming language in a VMS environment. The module dealing with programming gives a generic overview of the various programming steps. It is recommended that students enroll in a program–specific course to obtain the greatest benefit from a programming language.
- The use of commands or utilities that require privilege beyond the most basic privileges granted to users of your system.
- The use of commands that manipulate a multiprocessor environment.
- VMS programmed system services, common run–time library routines, or other features that require direct interaction with the operating system.
- Use of programmer productivity tools, databases, word processors, or any other optional software.
- References to and materials associated with VAXcluster systems.
- Advanced command procedure techniques, such as error handling, file I/O, dynamic arrays, CALL and GOSUB commands.
- Any functions, commands, or information related to "system management."
- File applications, specifically sorting records within a file and merging files.

EQUIPMENT/SPECIAL TOOLS

You will need almost constant access to any standard VAX configuration running VMS Version 5.0. The total laboratory time is approximately 20 hours.

COURSE RESOURCES

In addition to the VMS system itself, there are five major resources that should be available for you to complete this course:

1. This *Student Workbook*
2. The *manuals* of the VMS document set.
3. The *Course Administrator*, an individual at your site who monitors your progress, and ensures that the resources you require are available for your use.
4. *Online example files,* which support various exercises. Because this course is not computer–based, these files do not provide instruction independent of the Student Workbook.
5. The *Subject Matter Expert* is available to help you with technical questions, should you be unable to answer them using the course materials and references. In some cases, the subject matter expert and the course administrator are the same person.

Subdirectories of the course disk, whose logical name is DISK$V5UTLCOM_COURSE, store these example files. (Subdirectories are discussed in detail in the **Managing Files** module, Student Workbook – Volume II.) The subdirectory names are in the following format:

DISK$V5UTLCOM_COURSE:[COURSE.V5UTLCOM.mmm]

The "mmm" represents a subdirectory name, a three–letter code, for the particular module requiring files. Only two course modules require access to these subdirectories.

Table 1 contains a list of subdirectory names and the names of the modules they represent.

Table 1 Subdirectory and Module Names

Subdirectory Name	Module Name
COM	Writing Command Procedures
JOB	Submitting Batch and Print Jobs

To assist you in gaining access to the files in these subdirectories, a logical name has been created for each of them. The format of these logical names is V5UTLCOM$mmm, where "mmm" represents a subdirectory name listed in Table 1. (Logical names is a topic that is discussed in detail in the **Customizing the User Environment** module.)

Table 2 lists the logical name equivalences.

Table 2 Course Logical Names

Equivalence String	Logical Name
DISK$V5UTLCOM_COURSE:[COURSE.V5UTLCOM.COM]	V5UTLCOM$COM
DISK$V5UTLCOM_COURSE:[COURSE.V5UTLCOM.JOB]	V5UTLCOM$JOB

DOCUMENTATION

You should have access to the following manuals:

- This *Student Workbook*
- *Guide to VMS Files and Devices*
- *VMS DCL Dictionary*
- *Guide to Using VMS Command Procedures*
- *VMS Mail Utility*
- *VMS Phone Utility*
- *VAX EDT Reference Manual*
- *VAX Text Processing Utility Manual*
- *VMS DCL Concepts Manual*

One complete VMS documentation set must be available for reference.

Table 3 lists changes made to the VMS document set since the release of VMS Version 4.4.

Table 3 Changes to the VAX/VMS Document Set

V4.4 Manual	New Manual(s)
VAX/VMS DCL Dictionary	VMS DCL Dictionary
VAX/VMS Mail Utility Reference Manual	VMS Mail Utility Manual
VAX/VMS Phone Utility Reference Manual	VMS Phone Utility Manual
VAX/VMS DCL Concepts Manual	VMS DCL Concepts Manual
VAX Test Processing Utility Reference Manual	VAX Text Processing Utility Manual
VAX/VMS Text Processing Reference Manual	Guide to VMS Text Processing
VAX EDT Reference Manual	VAX EDT Reference Manual

COURSE ORGANIZATION

This *VMS Utilities and Commands* course is self–paced for independent study. It is divided into units, or modules, whose structure permits you to select those topics that are important to you. The course is designed to accommodate two groups of users: those with previous computer experience, and those with little or no experience. In general, the procedures that govern this course allow you to select the module you want to study, proceed at your own rate, and use as many or as few resources as you need or want.

Each module covers one or more of the nonprivileged operations a user typically performs at an interactive terminal. A module consists of one or more sections, each of which covers a convenient amount of material.

A module contains the following instructional elements:

- An *introduction*, which describes the purpose of the unit, provides some motivation for mastering its objectives, and outlines its contents.
- One or more *objectives*, which describe the operations for which the module provides instruction. Objectives are designed to focus your study efforts on a selected number of skills.
- A list of *resources* you may require to complete the unit. Some of these resources are distributed with this course; others are not. Since a complete document set is distributed with each VMS operating system, you should consult your course administrator or system manager for access to materials that do not come with this course.

- *Instructional materials* to introduce the skills described by the objectives. These include the following:

  — **Descriptive text** – The text of a module or section ties these various elements together. It describes the importance of a particular set of skills, listing the occasions when you are most likely to use them, or breaks a complex operation into a number of distinct steps. Always read the initial text of each lesson. Experienced users are encouraged to skip sections of text; inexperienced users are not.

  — **Figures** – Each figure summarizes a set of concepts or relationships presented by the module. Some figures help you visualize how commands you enter at your terminal affect your system. Others describe the syntax of command language elements or represent the steps the system follows when it executes a particular operation.

  — **Tables** – Each table summarizes the operations introduced by a particular section. Some tables help you to distinguish between related concepts. Like the figures, you can regard each table as a synopsis of written information presented in the lesson or covered by prescribed outside readings.

  — **Examples** – The examples illustrate how to use the concepts and operations a lesson describes. Many examples consist of a listing taken from a terminal session, and a line–by–line commentary. After reading the introduction to an example, attempt to read the listing. Refer to the commentary only if you do not understand the reason for using a particular command string. You may wish to reverse the process after reading the listing, skimming the commentary and referring only occasionally to the listing. Each session is designed so you can imitate it at your own terminal.

    Since the listings were made prior to the release of the final VMS software, you may occasionally find slight differences between your terminal displays and those of the listings.

  — **Written Exercises** – Written exercises provide practice in using concepts and interpreting various displays you generate at your terminal. They also allow you to practice picking an appropriate command to perform a specified operation. Each set of exercises is accompanied by **Solutions**. When appropriate, the solutions explain why a particular answer is correct.

  — **Laboratory Exercises** – Laboratory exercises provide practice in performing operations at a terminal. **Solutions** accompany each exercise.

Begin a new module by reading its introduction and objectives. If you can demonstrate mastery of the module objectives, take the test. Otherwise, work through the module at your own pace, completing readings, exercises, and activities in the recommended order.

In general, the figures and tables are designed to present the content of a lesson as quickly and efficiently as possible. If you are an experienced user, read these first, returning to the descriptive text only when you desire additional discussion. If you are an inexperienced user, read the supporting text, which you will find helpful and more thorough.

The most powerful instructional elements in the course are the examples and exercises. These provide you with actual operations and concepts in action. They also give you ample opportunity to practice. Regardless of your level of experience, you are more likely to master the objectives of a module by reading examples and performing exercises than by simply reading documents. It is suggested that you spend as much time as possible at a terminal.

COURSE MAP DESCRIPTION

The course map shows how each module of the course relates to the other modules and to the course as a whole. You can study a module in any sequence, provided you have met its prerequisites. Prerequisite modules are those whose arrows in the map point to the module you have selected. For example, you should not begin to study the **Communicating With Other Users** module until you have completed the **Creating and Editing Text Files** module.

After you have completely read this **About This Course** section, begin the course by studying the **Hardware and Software Overview** module.

You are free to study subsequent modules in any order, provided you respect the prerequisite relationship between modules in the map. You should, however, study only one module at a time.

COURSE GUIDELINES

You should be able to complete this *VMS Utilities and Commands* course in one to two weeks, provided you have easy access to an interactive terminal, and some familiarity with general–purpose operating systems like the VMS system. Follow the order of the modules as presented in the course map to take the complete course.

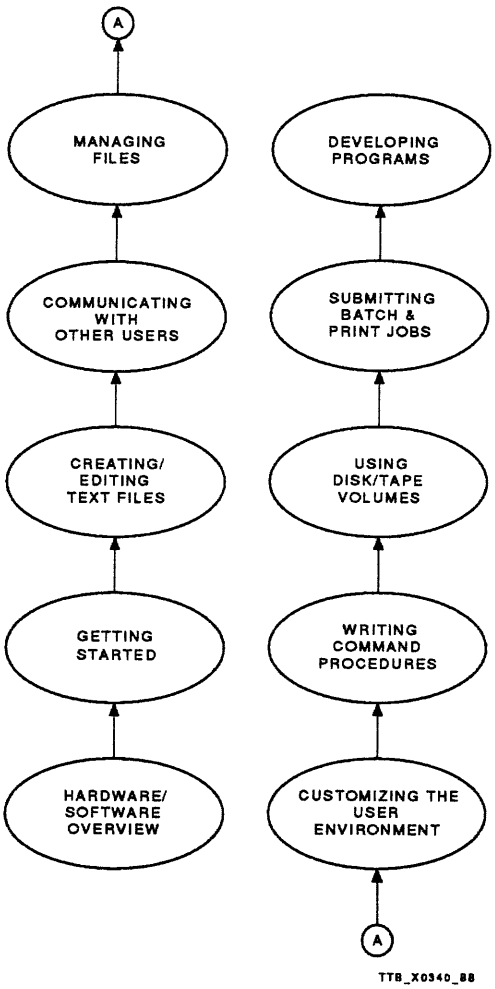


Figure 1 Course Map

COURSE CONVENTIONS

Table 4 describes the conventions used in this course.

Table 4 Course Conventions

Convention	Meaning
new terms, prompts	Terms that are introduced for the first time and system prompts are printed in italics.
CTRL/X	Press and hold the key labeled CTRL while you press another key (X). Many control key sequences have special meanings.
SHOW QUEUE	Names of commands in text are shown in uppercase and bold.
$ SHOW QUEUE/qualifier [queue–name]	Formats and command syntax are shown in bold. Words in uppercase are required, and words in lowercase represent elements that you must replace according to the description in the text.
`$ SHOW QUEUE/ALL_ENTRIES SYS$PRINT`	Actual examples of commands are shown in monospace type.
[ ]	Square brackets indicate that the enclosed item is optional. (Square brackets are not optional, however, in the syntax of some file specifications assignment statements.)
Type vs. Enter	When the word "type" is used in text, it means that you simply type a command. When the word "enter" is used, you must type the command and press the RETURN key.

HARDWARE AND SOFTWARE OVERVIEW

1.1 INTRODUCTION

When you begin work on a VMS system, you enter an environment consisting of devices, programs, and data. The devices that make up the physical computer are called *hardware*. The programs that control the hardware and process the data are called *software*. To perform job–related tasks on the system, you must use both the hardware and the software.

This module is designed to provide an introduction to VAX hardware, and an overview of the VMS software environment.

1.2 OBJECTIVES

To work on a VMS system, you should be able to:

- Identify the functions of each component of the hardware environment, namely:
  - — The Central Processing Unit (CPU)
  - — The console subsystem
  - — Main memory
  - — The input/output (I/O) subsystem
- Recognize the peripheral devices supported by VAX systems.
- State the definition of a system, and list the characteristics of the following types of configurations:
  - — Single processors
  - — Tightly coupled multiprocessors
  - — Networks

- Identify and describe the functions of each component of the software environment, namely:
  - — The operating system
  - — The command language interface
  - — Utilities
  - — Optional (layered) products
- Identify elements that make up a process in the VMS environment, such as:
  - — Process name
  - — Process identification
  - — Process type
  - — Privileges
  - — Quotas and limits

1.3 THE USER ENVIRONMENT

When you use your system interactively, your working environment consists of the following hardware and software elements:

- Your terminal, through which you communicate with your system.
- Your interactive process, which provides your working environment.
- The DIGITAL Command Language (DCL) – Command Language Interpreter (CLI), which translates instructions you enter and invokes the programs that execute them.
- The VMS operating system, which:
  - — Monitors your activities
  - — Handles all communications with physical devices
  - — Ensures that the resources of your system are shared appropriately among users
  - — Prepares programs to execute within the context of your process
- Other peripheral devices available for entering, storing, and printing data.
- The programs, utilities, and data available for your use.
- The additional VMS systems that may be available for your use.

1.4 COMPONENTS OF THE HARDWARE ENVIRONMENT

The VAX computer hardware is divided into four parts, or *subsystems*, each of which has a different function. Figure 1–1 shows the four subsystems. The subsystems represent the main functional areas of the VAX system.

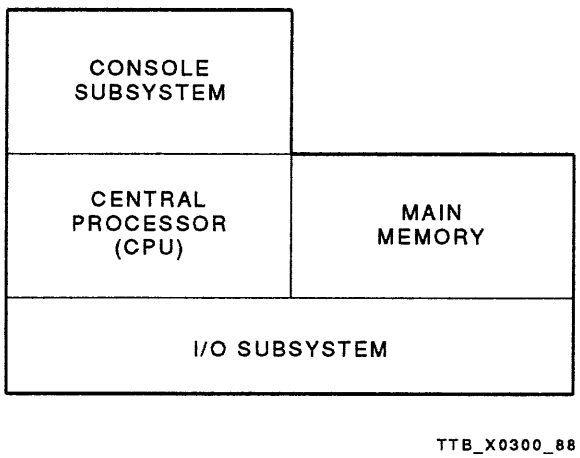


Figure 1–1 VAX Hardware Subsystems

1.4.1 The Central Processing Unit (CPU)

The primary function of the *CPU* is to execute instructions. It can only execute one instruction at a time. There are several different VAX CPUs, including the MicroVAX II processor, the VAX–11/780 processor, the VAX 8250 processor, and the VAX 8650 processor. The relative speed at which instructions are processed varies among VAX processors.

1.4.2 The Console Subsystem

The *console subsystem* communicates directly with the CPU. It is primarily used for:

- Starting up and shutting down the system
- Installing software
- Remote hardware diagnosis (if that option has been selected for your VAX system)

1.4.3 Main Memory

Main memory is used to store instructions and data. There are two types of memory on VAX systems:

- *READ ONLY memory (ROM)* stores part of the start–up sequence for the VAX system. The contents of ROM do not change.
- *READ/WRITE memory* stores data and instructions for programs that the CPU executes. The contents of Read/Write memory changes as different programs are executed. Read/Write memory is typically much larger than ROM.

1.4.4 Input/Output Subsystem

The *input/output subsystem* consists of devices that provide input to and output from the system. These devices are referred to as *peripherals*. Common peripherals include:

- Terminals
- Printers
- Disk drives
- Tape drives

Your interaction with the computer will be mostly with devices in the input/output subsystem.

1.5 PERIPHERAL DEVICES

Digital Equipment Corporation supports a variety of terminals, printers, and disk drives for use with VAX computers. The following sections discuss the general categories of peripherals.

1.5.1 Terminals

You can use a terminal to communicate with the computer. There are two types of terminals: *hardcopy* and *video*. Both types of terminals have keyboards for entering information. Hardcopy terminals print output on paper, while video terminals display output on a video screen. Figure 1–2 shows examples of hardcopy and video terminals.

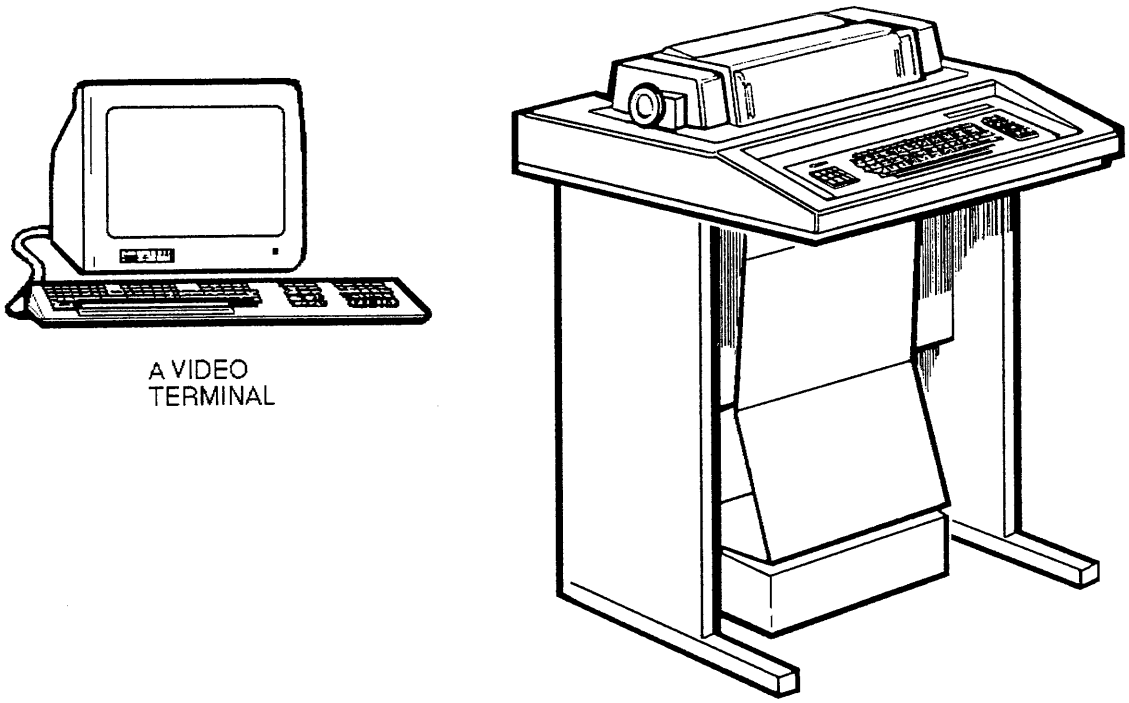


Figure 1–2 Sample Hardcopy and Video Terminals

1.5.2 Printers and Printer/Plotters

Printers provide output from the system. They come in a variety of sizes and types, for example:

- *Line printers* print one line of output at a time. Line printers are high–speed machines that are usually used for large quantities of output.
- *Letter–quality printers* produce letter–quality type. Several printwheels are available for use with these printers, to allow you to select a particular type style.
- *Laser printers* produce high–quality print and graphics. Several different typefaces, or *fonts*, are available with laser printers.

Printer/plotters are used for graphic output. They are able to produce textual output and plot graphic representations of data. Figure 1–3 shows examples of each type of printer and a printer/plotter.

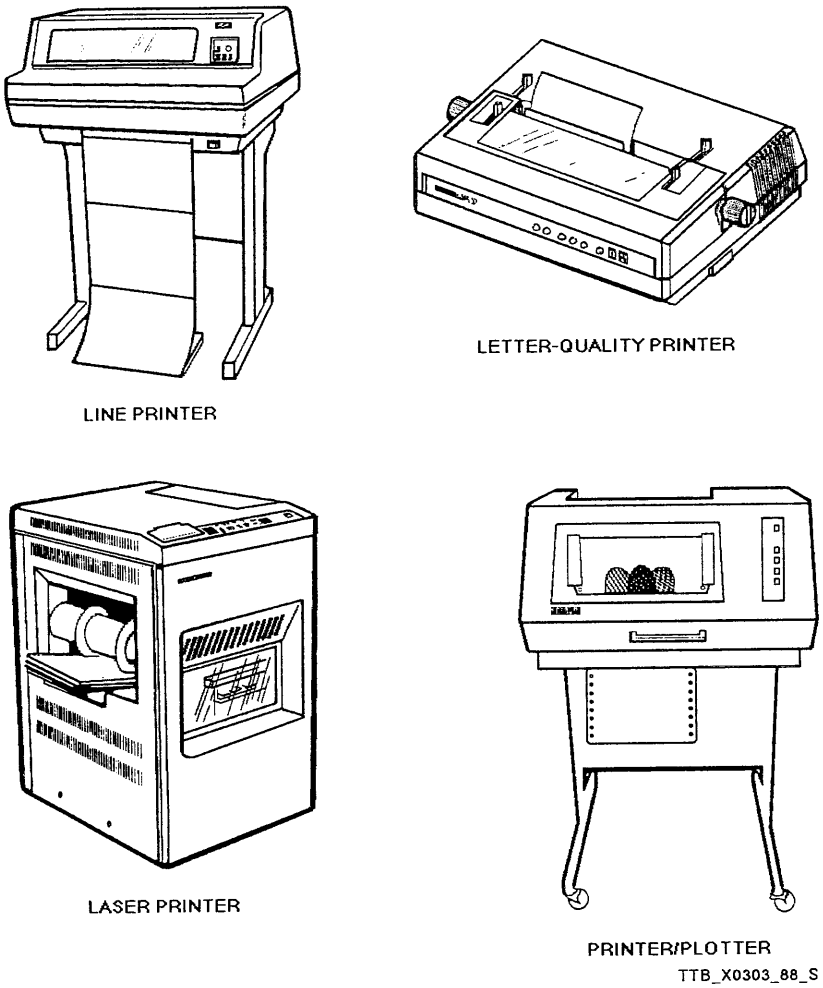


Figure 1–3 Sample Printers and Printer/Plotter

1.5.3 Disk and Tape Drives

Disk and tape drives are devices that record and read data on magnetic disks and tapes. These drives are sometimes called *mass storage devices*, while the disks and tapes used in the drives are commonly referred to as the *storage media*. Disks usually store frequently used data. Disks are either removable or fixed. Tapes usually store backup copies of data, and infrequently used data.

Data on a disk or tape is divided into *files*. Each file has a name and other identifying information. Several of the modules in this course teach you how to create and manipulate your own files.

Disks vary in size, shape, and capacity. The various types of removable disks include:

- Cartridges
- Disk packs
- Diskettes
- CDROM

Figure 1–4 shows examples of various types of disks. Each type of disk is used with a particular disk drive. Figure 1–5 shows some examples of disk drives.

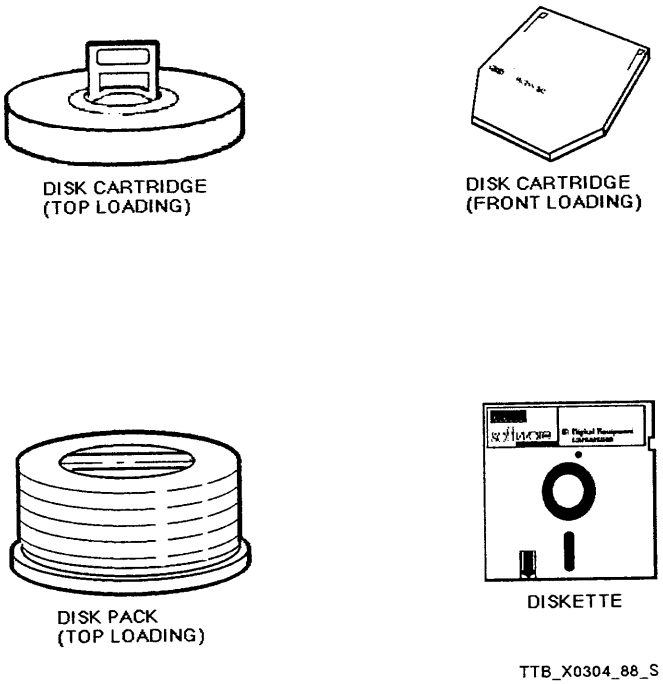


Figure 1–4 Examples of Disks

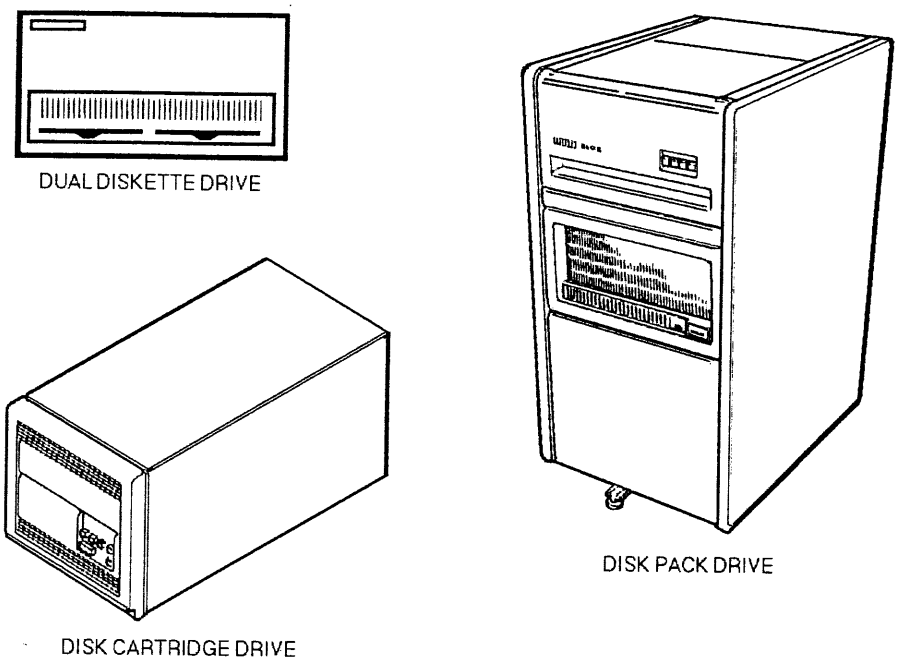


Figure 1–5 Examples of Disk Drives

Just as there are different disks for use on VAX systems, there are also different tapes. Tapes also vary in size, shape, and capacity. There are two types of tape media:

- Reel tapes
- Tape cartridges

Figure 1–6 shows examples of tapes, and Figure 1–7 shows some tape drives.

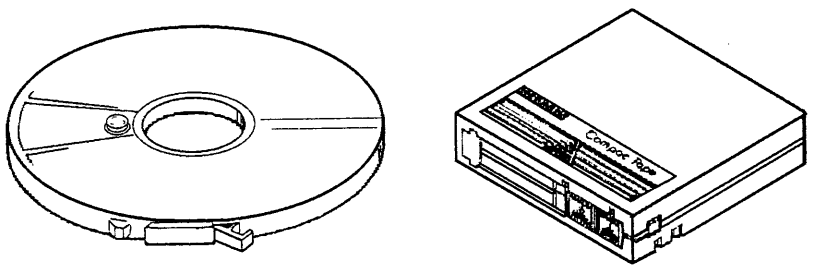


Figure 1–6 Examples of Tape Media

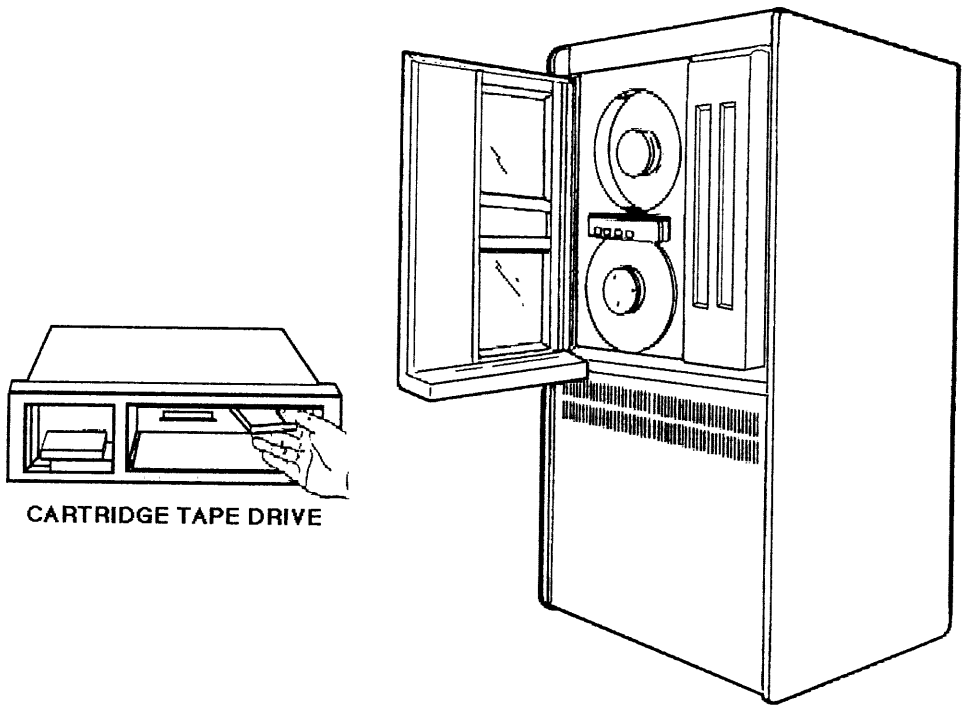


Figure 1–7 Sample Tape Drives

1.6 SYSTEM CONFIGURATIONS

Given the various VAX processors and peripheral devices, many different configurations can be built. System configurations can be classified as:

- *Single processors*
- *Multiple–processor configurations*

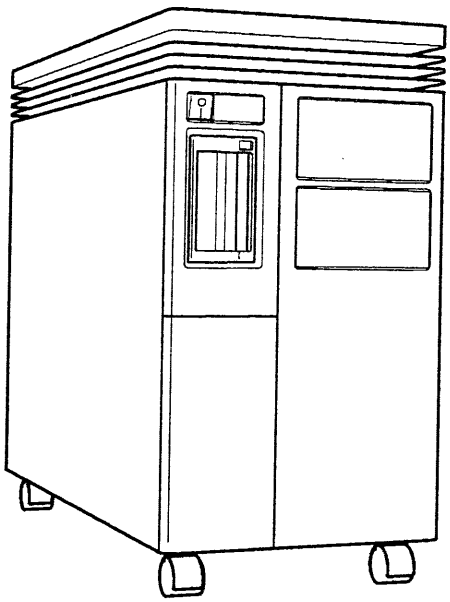
In this context, the word "system" can have different meanings. A single VAX processor, including its peripheral devices, is called a system. However, the word "system" is also used to reference a collection of VAX processors that make up a multiple processor configuration. The following sections describe the different types of system configurations.

1.6.1 Single–Processor Configurations

A *single–processor configuration* is any single VAX processor and its peripheral devices. The family of VAX processors includes:

- VAX 8810
- VAX 8650
- VAX 8250
- VAX–11/785
- VAX–11/780
- MicroVAX 3000
- MicroVAX II

Note that this list reflects only some of the actively marketed processors that are available at this time. Figure 1–8 shows a MicroVAX II processor, and Figure 1–9 shows a VAX 8600 processor.



TTB_X0308_88

Figure 1–8 MicroVAX II Processor

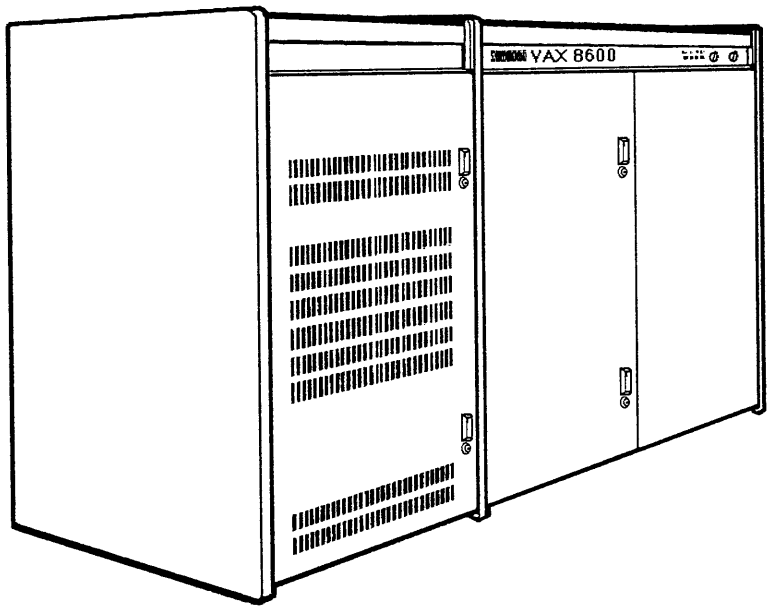


Figure 1–9 VAX 8600 Processor

1.6.2 Multiple–Processor Configurations

A *multiple–processor configuration* consists of two or more communicating processors. There are three classifications for multiple–processor configurations:

- Tightly coupled multiprocessors
- Networks
- VAXcluster systems

In tightly coupled configurations, the processors share the same operating system code. They cannot operate independently.

In a network, each processor executes a separate copy of the operating system. They can operate independently. The ability of systems in a network to operate independently is a major asset for systems requiring high availability.

VAXcluster systems are positioned midway between tightly coupled configurations and networks. Each VAX processor, or *node*, in a cluster executes its own copy of the VMS operating system. A cluster can be configured so that users on all nodes use a common file system and common devices. A cluster can continue operating without the participation of all processors.

1.6.2.1 Tightly Coupled Configurations

The VAX 8820 system is an example of a *tightly coupled configuration.* It consists of two CPUs sharing memory via a system bus. (A system bus is an interconnect device that is used to connect memory and peripherals to a VAX processor.)

Tightly coupled systems provide high performance. They are used in compute–intensive applications, such as simulation and statistical processing.

A tightly coupled system is operated and managed as a single domain. The attached processor is transparent to the system users. To the user, a tightly coupled system appears the same as a single–processor configuration. Figure 1–10 shows an example of a tightly coupled system configuration.

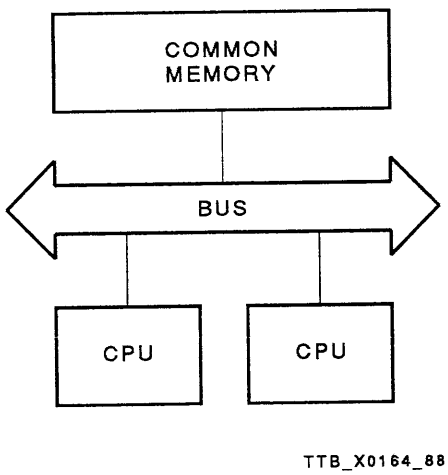


Figure 1–10 A Tightly Coupled System Configuration

1.6.2.2 Networks

A *network* can consist of two or more communicating processors. A VMS system can be connected to other DIGITAL systems, or to other manufacturers' systems, to establish a network. This section discusses DIGITAL–to–DIGITAL networks, which consist of two or more DIGITAL systems.

When you establish a network in a limited geographical area, such as in a building, it is called a *local area network*, or *LAN*. A network that spans a larger area, such as several cities or several countries, is called a *wide area network*.

DIGITAL–to–DIGITAL networks are established using:

- Two or more processors
- Hardware communication devices
- Data transmission media
- Terminal servers (optional)
- DECnet software

DECnet software enables communication between the networked systems. This communication allows a user who is logged in to one system to communicate with users on other systems and to access disk files on other systems. Users can also employ programs as the means of communication between systems. Figure 1–11 shows an example of a DECnet network.

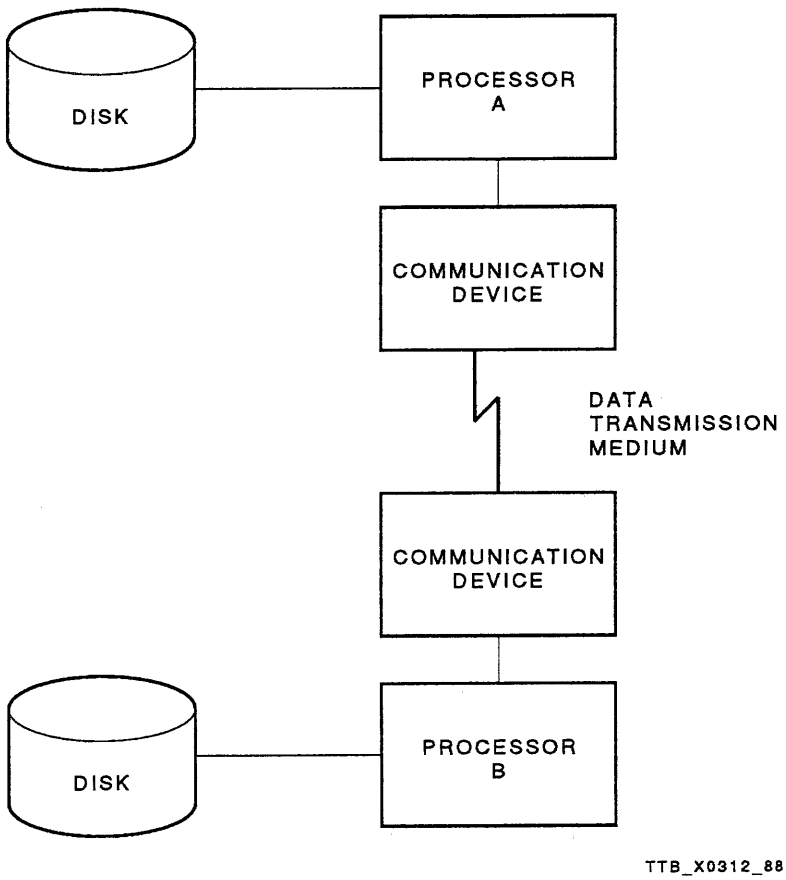


Figure 1–11 A DECnet Network

Notes on Figure 1–11:

1. This network consists of two processors, or nodes. Each node has a disk drive.
2. Each processor in the network has an attached communication device. The communication devices are connected by a data transmission medium.
3. Access to disk files stored on a given node depend upon the availability of that node. For example, if Processor A is shut down, any disk files stored on Processor A's disk become inaccessible to users logged in to Processor B.

1.6.2.3 VAXcluster Systems

A *VAXcluster system* is a flexible multiprocessing system. Like a tightly coupled system, a VAXcluster system is managed as a single domain. The nodes in a VAXcluster system can share disk and tape devices and a common file system. A VAXcluster system can be configured to present an identical user environment on every node.

However, each VMS system in a properly configured VAXcluster system boots and fails separately, as in a network.

VAXcluster systems allow the user to perform functions similar to a network. The important difference between a VAXcluster system and a network is that, in addition to providing the functions of a network, VAXcluster systems provide:

- Higher availability of system resources.
- Faster and easier sharing of information and resources between nodes.

You can form a VAXcluster system from just a few components and expand it as your computing needs increase. A VAXcluster system may start as one or two VMS systems connected to a Computer Interconnect (CI) or Ethernet. Each system can have its own local peripherals, including mass storage devices, printers, and local terminals.

As the need for computing power increases, you can increase the number of VMS systems in your VAXcluster system. As your mass storage needs increase, you can add more storage devices. Figure 1–12 shows a simple VAXcluster system and its expansion into a larger VAXcluster system.

A VAXcluster system may have hardware similar to a network. It may contain the same components as a network, such as:

- VAX processors
- Communication devices
- Transmission media
- DECnet software

A VAXcluster system may also have other specific hardware, such as a *Hierarchical Storage Controller (HSC)* or a *Computer Interconnect (CI).*

The major difference between a VAXcluster system and a network is VMS cluster software, which synchronizes access to shared resources.

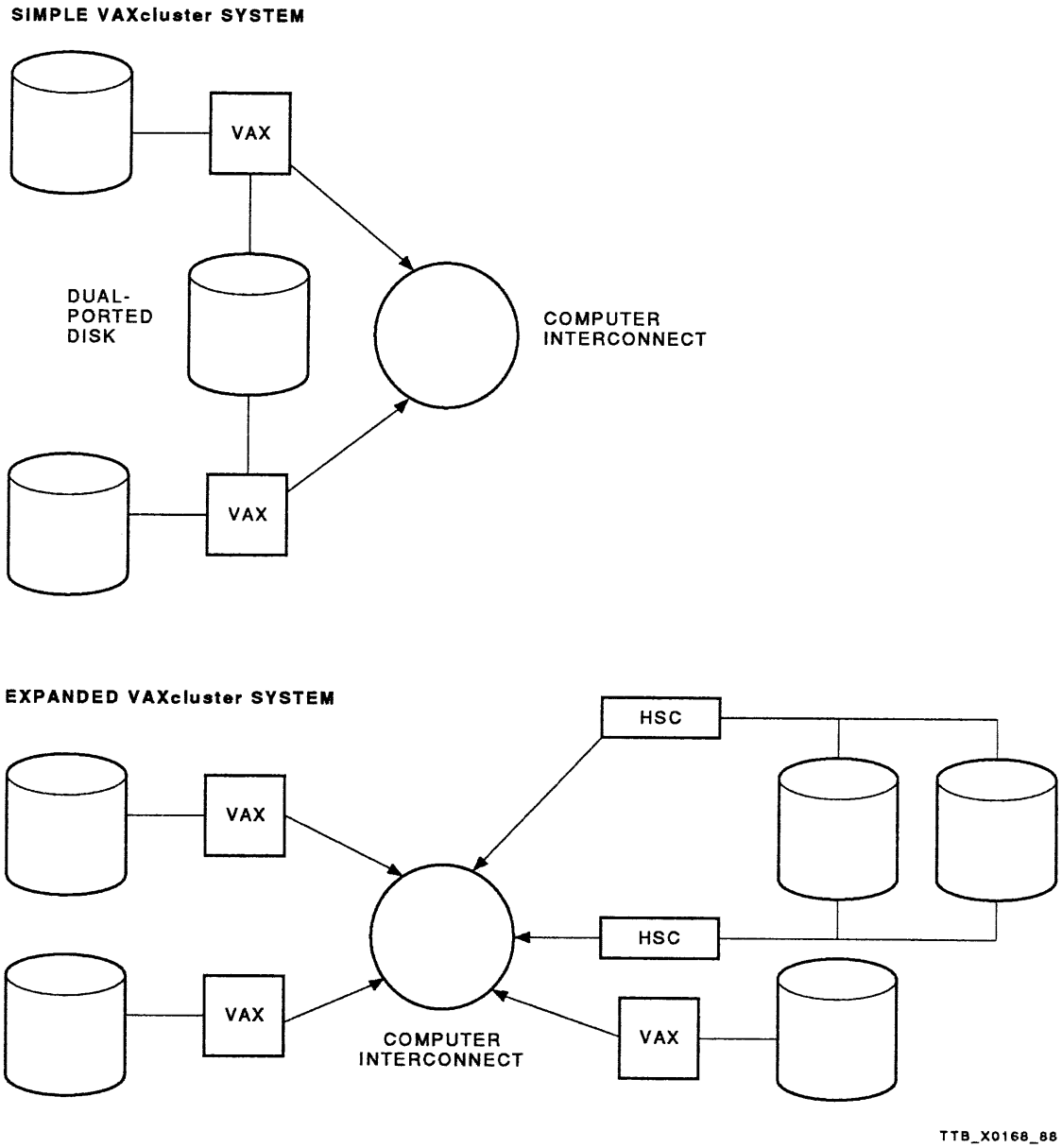


Figure 1–12 VAXcluster System Structure

Notes on Figure 1–12:

1. The expanded VAXcluster system has added a third VAX system (with a local disk), and two HSC controllers with two dual–ported disks connected to them.
2. Any of the three VAX systems in the expanded VAXcluster system can mount the two disks that are connected to the HSC controllers. HSC disks are more available to users than local disks, because HSC disks are not dependent upon the availability of any processor.
3. The disks are dual–pathed to the HSC controllers, further increasing the disks' availability in the VAXcluster system. If one HSC fails, all traffic to connected disks automatically switches to the second HSC.

1.7 THE VMS OPERATING SYSTEM

An *operating system* is a collection of programs that control the operations and manage the resources of a computer system. On a VMS system, this collection of programs is called the *VMS operating system.*

Like any operating system, the VMS operating system performs three major functions. It:

- Provides the means for you to communicate with the hardware devices that make up your system.
- Creates a working environment in which you can access the resources needed to perform your tasks without interfering with other system users' activities.
- Schedules the use of the CPU, physical memory, and peripheral devices to provide equitable access for all users, while using these resources as efficiently as possible.

The programs that make up the VMS operating system are sometimes called the system *software.* Typical activities of the system software include:

- Loading programs and data into memory from storage
- Scheduling the order of action by the CPU
- Allocation of resources, such as physical memory
- Input and output (I/O) to other devices

1.7.1 DIGITAL Command Language

The user interface is the means by which a user communicates with the operating system. On a VMS system, the user interface is the *DIGITAL Command Language (DCL).*

Other operating systems may use obscure command names or complex command syntax. DCL uses common words for its commands and qualifiers. This use of common words makes it easier to:

- Remember DCL commands
- Enter DCL commands at a terminal
- Recognize and correct syntax errors

DCL commands can be used to:

- Perform file manipulation tasks, such as copying, renaming, and deleting files, or displaying directories.
- Display information about the status of the system, other users on the system, devices connected to the system, available resources, and so on.
- Execute system utilities or run user–written programs.

The dollar sign ($) is the default DCL command prompt. In other words, when the system displays the dollar sign at your terminal screen, it is ready to accept a DCL command. You enter a DCL command by typing in the command following the dollar sign, and pressing the RETURN key.

When you enter a DCL command, it is translated and interpreted by the Command Language Interpreter (CLI). If the command you enter is correct, the CLI then calls those VMS system routines needed to perform the command.

DCL commands and qualifiers can be abbreviated, and each abbreviation must be unique. Abbreviating most DCL commands and qualifiers to four characters is enough to make them unique.

1.7.2 Utilities

Utilities are software tools provided with the VMS software that perform specified tasks. Some utilities are invoked as DCL commands, by entering command verbs and qualifiers at the DCL prompt. Other utilities may have their own set of commands, and may use their own command prompt.

The utilities provided on a VMS system perform a wide variety of tasks. Some of the utilities available on a VMS system include:

- Text editors
  - — EDT Editor
  - — Extensible VAX Editor (EVE)
- Text Processors
  - — DIGITAL Standard Runoff (DSR)
- Debugging and Programming Tools
  - — VMS Debugger
- Communication Utilities
  - — MAIL
  - — PHONE

1.7.3 Optional (Layered) Products

In addition to the software provided with the VMS system, you may need to use additional software products to perform specific tasks. Since not every user may need these software products, they are optional. Layered products are so named because they provide an additional layer of features above the ones provided with the VMS operating system.

Types of optional (layered) products include:

- *Language compilers*, which translate programs written in a specific programming language, such as FORTRAN or Pascal, into a format the system can understand.
- *Communications software*, such as DECnet, which allows one VMS system to communicate with another VMS system, or another system manufactured by Digital Equipment Corporation or another vendor.
- *Diagnostic software*, which locates hardware and software problems on the system.
- *Data management tools* such as DATATRIEVE, which allow the creation, maintenance, and use of databases.

1.8 THE WORKING ENVIRONMENT

1.8.1 The Process

The working environment on a VMS system is defined in terms of the *process*. All work on a system is performed within a process. The system uses processes to determine the order in which jobs are executed, the availability of system resources for a particular job, and the allocation of those resources.

A process has four components:

1. *The Hardware Context*—Values contained in the processor registers that describe what the process is doing.
2. *The Software Context*—Controls what the program is allowed to do.
3. *Virtual Address Space*—Process addresses are mapped to physical addresses.
4. *The Image*—The image, or program, that contains instructions to do the actual work.

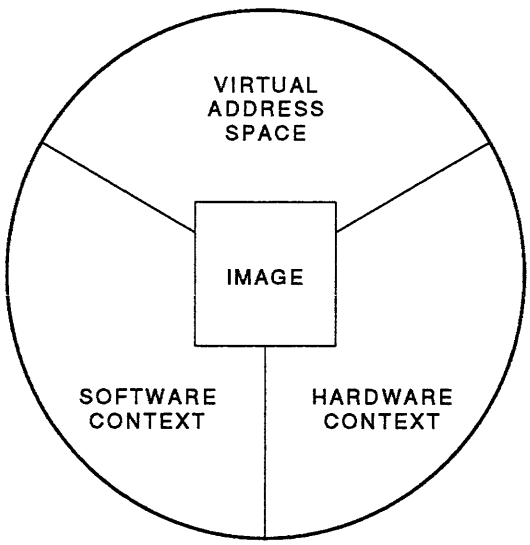


Figure 1–13 Components of a Process

1.8.2 Process Types

- *Interactive Process*

  Of the several types of processes on a VMS system, the most common is the interactive process. This interactive process is created when a user logs in to a system at a terminal. If the user name and password are valid, the system creates a process. The **Getting Started** module of this course teaches you how to log in and use an interactive process.

- *Subprocess*

  You can issue certain DCL commands that create a subprocess. A subprocess is a process that uses some of the same resources as the parent process. For example, you can run a program, then create a subprocess and enter commands using the subprocess, without having to terminate the program that is running.

- *Batch Process*

  Another common type of process is the batch process. The system software creates a batch process by executing a file, called a *command procedure*, that contains DCL commands. Before the system can execute this command procedure, you must place the file into a batch queue.

  When the system schedules the execution of the command procedure, it creates a batch process, which automatically executes the commands in the file. Advantages of using batch processes include the ability to delay the creation of the batch process, as well as the fact that your terminal is free for other work. The **Submitting Batch and Print Jobs** module of this course instructs you on how to create a batch process.

1.8.3 The System User Authorization File

Much of the information the system uses to create a process comes from the *System User Authorization File (UAF)*. This file contains information about all users who are authorized to obtain access to the system. The information about each user is usually placed in this file by the *system manager*, who is the person in charge of maintaining the operating system.

Information in the System User Authorization File includes:

- *User Identification Code (UIC).* The UIC is used by the system to determine the owner of files on a storage device, and to determine if a user can access a particular file on a VMS system. The UIC is made up of a group number and a user number. Usually each user on a system has a unique UIC.
- *Default directory.* The default directory determines what disk and directory are used by the process as its default file storage area.
- *Privileges.* Some actions on a VMS system are protected; not everyone can perform them. The privileges associated with a process determine whether or not the user can perform these tasks. If the process has the required privilege, then the user can perform the action. Otherwise, a system message is displayed.
- *Priority.* The process priority is used by the system to determine the order in which work is executed on the system. Timesharing process priorities run from 1 to 15. The higher the priority value, the sooner the job is run.
- *Quotas and Limits.* The quotas and limits assigned to a process determine which system resources are provided to the process and in what quantity.

1.9 SUMMARY

- There are four main functional subsystems of VAX computers:
  - — The CPU – executes instructions.
  - — The console subsystem – communicates with the CPU to monitor and control the system.
  - — Main memory – stores data and instructions.
  - — The I/O subsystem – consists of devices that provide input to and produce output from the system. These devices are referred to as peripherals. Peripherals include terminals, printers, disk drives, and tape drives.

The software environment is made up of several components:

- The VMS Operating System:
  - — Controls software on the system.
  - — Provides the means of communication with other hardware devices on the system.
  - — Schedules the allocation of resources and the execution of programs.
- The user interface with the VMS system is the DIGITAL Command Language (DCL):
  - — The means by which a user communicates with the system.
  - — Uses common English–like words.
  - — Interpreted by the Command Language Interface (CLI).
- Utilities are software tools that perform specific tasks:
  - — Provided with the system software.
  - — Include tools such as editors, text formatters, and communication utilities.
- Optional Layered Products:
  - — Perform tasks beyond those of the system software.
  - — Must be purchased and installed separately.

The working environment is defined in terms of a process:

- The system creates and controls processes.
- Information used to create processes is stored in the System User Authorization File (SYSUAF).

- VAX system configurations can be classified as single processors and multiple–processor configurations.
- A single–processor configuration is any single VAX processor and its peripheral devices.
- There are three types of multiple–processor configurations:
  - — Tightly coupled multiprocessors
  - — Networks
  - — VAXcluster systems—configured midway between multiprocessors and networks
- Multiple–processor configurations consist of:
  - — Processors
  - — Peripheral devices
  - — Communication devices
  - — Transmission media
  - — Terminal servers (optional)
- A local area network spans a limited geographical area.
- A wide area network spans a larger area.
- The important difference between a network and a VAXcluster system is that the sharing of information between nodes is much faster and easier in a VAXcluster system.

1.10 WRITTEN EXERCISE I

In the exercise below, match each description with the appropriate component of the hardware environment. Components of the hardware environment may be used once, more than once, or not at all.

Components of Hardware Environment

a. CPU

b. Console Subsystem

c. Main Memory

d. I/O Subsystem

Descriptions

1. _____ Stores instructions and data

2. _____ Used to monitor and control the system

3. _____ Consists of peripherals

4. _____ Executes instructions

5. _____ Used for starting up and shutting down the system

1.11 WRITTEN EXERCISE II

Write the letter of the term that best completes each of the following statements.

1. _____ are used to connect the various subsystems of the computer.
   a. Peripheral devices
   b. Network communication devices
   c. Interconnect devices
   d. Storage devices
2. _____ have a screen for displaying information.
   a. Hardcopy terminals
   b. Video terminals
   c. Laser printers
   d. Mass storage devices
3. _____ is NOT a peripheral device.
   a. Terminal
   b. Printer
   c. CPU
   d. Disk drive
4. _____ are high–speed machines that are usually used for large quantities of stored output.
   a. Hardcopy terminals
   b. Disk drives
   c. Laser printers
   d. Line printers

5. _____ is NOT a type of disk.

   a. Reel

   b. Cartridge

   c. Diskette

   d. Disk pack

6. _____ record data on magnetic media.

   a. Disk drives

   b. Tape drives

   c. Terminal servers

   d. VAXcluster systems

7. A _____ is a network established within a limited geographical area.

   a. Wide area network

   b. Local area network

   c. Multiprocessor

8. In addition to providing the functions of a network, _____ also provide higher availability of system resources, and faster and easier sharing of information and resources.

   a. VAXcluster systems

   b. Wide area networks

   c. Tightly coupled multiprocessors

   d. Terminal servers

9. The major difference between a VAXcluster system and a network is _____.

   a. VMS DECnet software

   b. VMS cluster software

   c. The communication devices

   d. The processors

1.12 WRITTEN EXERCISE III

Example 1–1 displays the characteristics of a privileged process on your system. Using the information displayed in the example, determine the value of each of the following parameters:

1. _____ Account name
2. _____ Default Device and Directory Specification
3. _____ Interactive Terminal Specification
4. _____ Password
5. _____ Process Identification Code
6. _____ Process Name
7. _____ User Identification Code
8. _____ User Name
9. _____ Priority

Privileges (list them)

10. _____
11. _____
12. _____
13. _____ CPU Limit
14. _____ Open File Quota
15. _____ Subprocess Quota

1.13 WRITTEN EXERCISE III (CONT)

```
31-DEC-1987 13:45:39.54   VTA15:              User: SMITH
Pid: 20400140   Proc. name: SMITH           UIC: [TRNG,SMITH]
Priority:   4   Default file spec: DISK:[SMITH]
Devices allocated: VTA15:
Process Quotas:
Account name: TRNG
CPU limit:                          Infinite  Direct I/O limit:          18
Buffered I/O byte count quota:         12192  Buffered I/O limit:        18
Timer queue entry quota:                  10  Open file quota:           63
Paging file quota:                      9063  Subprocess quota:           2
Default page fault cluster:               64  AST limit:                 22
Enqueue quota:                            40  Shared file limit:          0
Max detached processes:                    1  Max active jobs:            0

Accounting information:
Buffered I/O count:       21298  Peak working set size:       1500
Direct I/O count:         11639  Peak virtual size:           1789
Page faults:              26172  Mounted volumes:                0
Images activated:           112
Elapsed CPU time:       0 00:11:33.90
Connect time:           0 04:58:58.78

Process privileges:
TMPMBX                 may create temporary mailbox
NETMBX                 may create network device
GROUP                  may affect other processes in same group
Process rights identifiers:
INTERACTIVE
LOCAL
TRNG
SYS$NODE_SUPER

Process Dynamic Memory Area
Current Size (bytes)          25600    Current Total Size (pages)      50
Free Space (bytes)            21184    Space in Use (bytes)          4416
Size of Largest Block         21072    Size of Smallest Block          56
Number of Free Blocks             3    Free Blocks LEQU 32 Bytes        0
```

Example 1–1 Process Parameters of a Sample Interactive Process

1.14 WRITTEN EXERCISE IV

Match each of the following operations with the parameter that controls your ability to perform it. Some operations are controlled by more than one parameter.

Parameters

a. Password

b. Priority

c. Privilege

d. Process Identification Number (PID)

e. Resource Limit

f. User Identification Code (UIC)

g. User Name

Operations

1. _____ Logging in to your system

2. _____ Deleting a file that belongs to another user

3. _____ Opening a large number of files

1.15 WRITTEN EXERCISE I—SOLUTIONS

In the exercise below, match each description with the appropriate component of the hardware environment. Components of the hardware environment may be used once, more than once, or not at all.

Components of Hardware Environment

a. CPU

b. Console Subsystem

c. Main Memory

d. I/O Subsystem

Descriptions

1. c,d Stores instructions and data
2. b Used to monitor and control the system
3. d Consists of peripherals
4. a Executes instructions
5. b Used for starting up and shutting down the system

1.16 WRITTEN EXERCISE II—SOLUTIONS

Write the letter of the term that best completes each of the following statements.

1. __c__ are used to connect the various subsystems of the computer.
   a. Peripheral devices
   b. Network communication devices
   c. Interconnect devices
   d. Storage devices
2. __b__ have a screen for displaying information.
   a. Hardcopy terminals
   b. Video terminals
   c. Laser printers
   d. Mass storage devices
3. __c__ is NOT a peripheral device.
   a. Terminal
   b. Printer
   c. CPU
   d. Disk drive
4. __d__ are high–speed machines that are usually used for large quantities of stored output.
   a. Hardcopy terminals
   b. Disk drives
   c. Laser printers
   d. Line printers

5. __a__ is NOT a type of disk.

   a. Reel

   b. Cartridge

   c. Diskette

   d. Disk pack

6. __b__ record data on magnetic media.

   a. Disk drives

   b. Tape drives

   c. Terminal servers

   d. VAXcluster systems

7. A __c__ is a network established within a limited geographical area.

   a. Wide area network

   b. Local area network

   c. Multiprocessor

8. In addition to providing the functions of a network, __a__ also provide higher availability of system resources, and faster and easier sharing of information and resources.

   a. VAXcluster systems

   b. Wide area networks

   c. Tightly coupled multiprocessors

   d. Terminal servers

9. The major difference between a VAXcluster system and a network is __b__.

   a. VMS DECnet software

   b. VMS cluster software

   c. The communication devices

   d. The processors

1.17 WRITTEN EXERCISE III—SOLUTIONS

Compare your answers with those shown below.

1. TRNG Account name
2. DISK:[SMITH] Default Device and Directory Specification
3. VTA15: Interactive Terminal Specification
4. Not Displayed Password
5. 20400140 Process Identification Code
6. SMITH Process Name
7. [TRNG,SMITH] User Identification Code
8. SMITH User Name
9. 4 Priority

   Privileges (list them)
10. GROUP
11. TMPMBX
12. NETMBX
13. Infinite CPU Limit
14. 63 Open File Quota
15. 2 Subprocess Quota

1.18 WRITTEN EXERCISE IV—SOLUTIONS

Match each of the following operations with the parameter that controls your ability to perform it. Some operations are controlled by more than one parameter.

Parameters

a. Password

b. Priority

c. Privilege

d. Process Identification Number (PID)

e. Resource Limit

f. User Identification Code (UIC)

g. User Name

Operations

1. g,a Logging in to your system

   To log in to a system, you must know the user name of a record in the UAF. You must also know the password that corresponds to that user name.

2. c,f Deleting a file that belongs to another user

   Your ability to delete a file depends on your UIC.

3. e Opening a large number of files

   A resource limit (FILLM) determines the number of files you can open simultaneously.

GETTING STARTED

2.1 INTRODUCTION

To perform daily tasks on a VMS system, you must issue instructions written in the DIGITAL Command Language (DCL). The Command Language Interpreter (CLI) of DCL analyzes your requests. Like any language, DCL consists of a vocabulary and rules of grammar.

The vocabulary of DCL includes commands, parameters, and qualifiers, all of which perform functions similar to those of verbs, nouns, adverbs, and adjectives in English. When you arrange them to form a command line, the VMS system knows what operations to perform.

DCL is one of the simplest languages ever developed for a DIGITAL computer system. Behind this simplicity, however, lies a capacity for complex communication.

This module introduces you to:

- Communicating with your VMS system using the DIGITAL Command Language (DCL)
- Using both online and printed VMS documentation

2.2 OBJECTIVES

To effectively use the interactive features of the VMS system, you should be able to:

- Log in and log out of the system.
- Use the DIGITAL Command Language (DCL) to make VMS perform jobs.
- Use the VMS Help facility and VMS documentation to obtain information about DCL commands and error messages.
- Interpret any VMS error messages and issue a corrected command by using the DCL command–line editor.
- Obtain and interpret information about the terminal, process, and system.

2.3 RESOURCES

To complete this module, you should have access to the following documents:

- *VMS General User's Manual*
- *VMS DCL Dictionary*

2.4 USER NAME AND PASSWORD

Your *user name* consists of 1 to 12 characters. Your user name is usually your last name and is assigned by the system manager.

Your password consists of 1 to 31 characters. Legal characters are A through Z (both upper– and lowercase), numerals 0 through 9, the dollar sign ($), and the underscore (_). Your password is usually assigned by your system manager, but you are allowed to change it at any time.

2.5 BEGINNING AND ENDING A TERMINAL SESSION

To access your system at a terminal, and to identify yourself, you must log in. There must be a record for you in the User Authorization File (UAF) before you can log in. The system manager usually establishes this record, which provides you with a user name and a password.

If the user name and password you enter match those in the system UAF record, the system creates an interactive process for your use, and displays an informational message, followed by the default DCL prompt ($). If your user name and password do not match your UAF record, the system displays an error message and denies you system access. In the event you are denied access to the system, try to log in again. Notify your system manager if you continue to have trouble.

To log in to the system:

- Turn on your terminal.
  If your terminal has a REMOTE/LOCK switch, set the switch to REMOTE.
- Press the RETURN key on the terminal keyboard.
- In response to the prompt *Username:* type your user name, then press RETURN.
- In response to the prompt *Password:* type your password, then press RETURN. The system does not display your password.

To log out of the system:

- At the default DCL prompt ($), type **LOGOUT** and press RETURN.

Figure 2–1 shows the sequence of steps to log in to a VMS system.

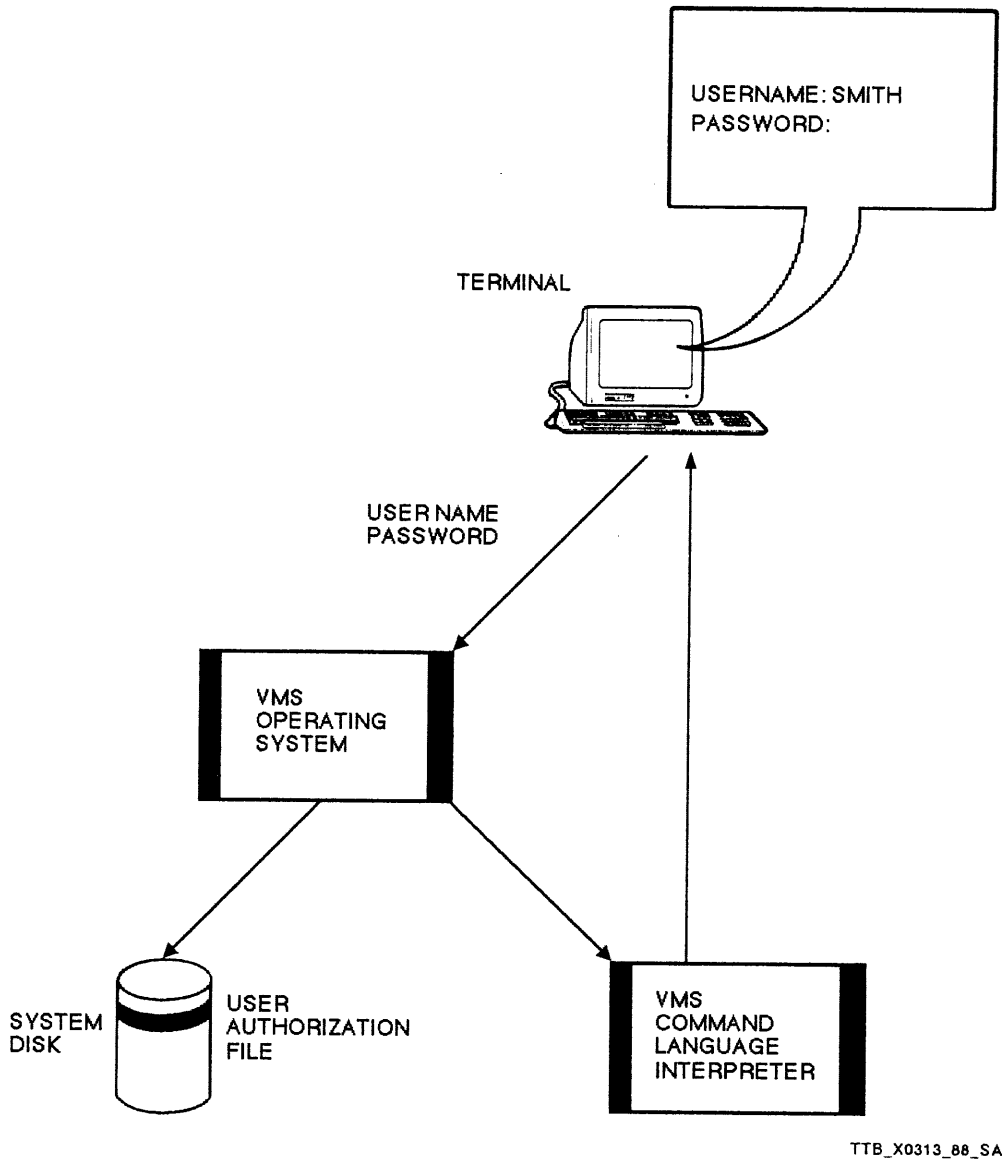


Figure 2–1 Enter a Valid Name and Password

```
VAX/VMS SUPER
Username: SMITH
Password:
Welcome to VAX/VMS SUPER
Last interactive login on Wednesday, 30-DEC-1987 10:27

$ LOGOUT/FULL
SMITH      logged out at  5-JAN-1988 10:53:51.92
Accounting information:
Buffered I/O count:           46      Peak working set size:    333
Direct I/O count:             24      Peak page file size:     1969
Page faults:                 496      Mounted volumes:            0
Charged CPU time:     0 00:00:02.51   Elapsed time:     0 00:00:18.42
```

Example 2–1 How to Log In and Log Out

NOTE
The /FULL qualifier of the LOGOUT command displays a summary of the accounting information for the terminal session. LOGOUT does not display any information.

2.6 DCL COMMAND FORMAT

Figure 2–2 illustrates the elements of a DCL command specification. Tables 2–1 and 2–2 review the elements you can use to make up an instruction.

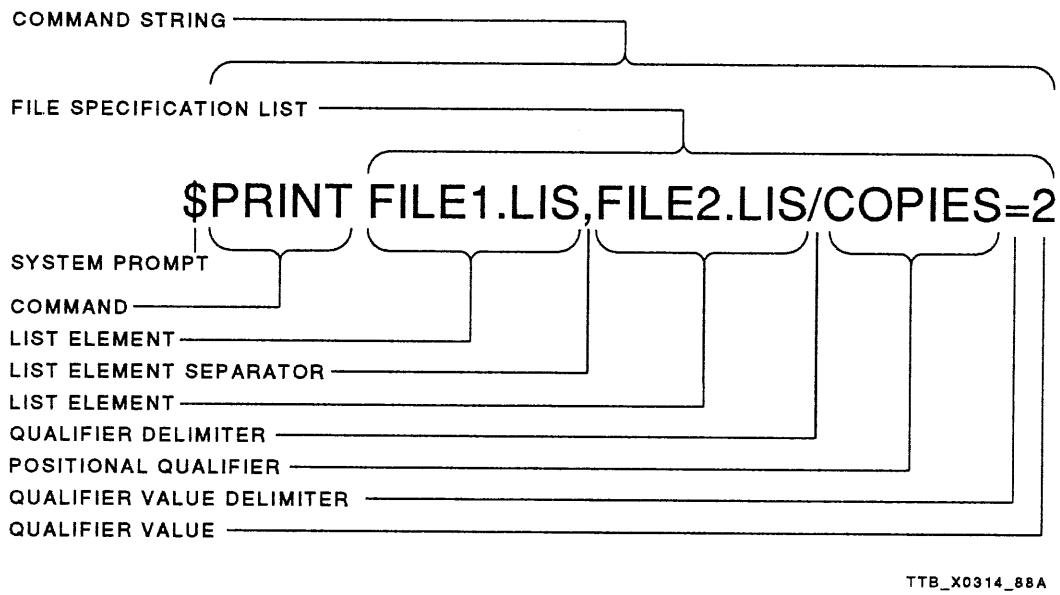


Figure 2–2 The Elements of a Command Line

2.6.1 Command Line Construction

- One or more spaces or tabs separate commands, command options, and parameters from each other.
- Slash marks (/) separate qualifiers from commands and parameters.
- Commas (,) separate each element of a file specification list.
- A file qualifier following a command affects each element in a file specification list. A file qualifier following a particular file affects only that file.
- Pressing RETURN passes the command line to the DCL CLI for execution regardless of the cursor position on the line.

Table 2–1 Elements of a DCL Command Line

Command Element	Comments
Command Line	A command line is the complete specification of a DCL command. Although you can continue a command on more than one line, the term command line defines the entire command that you pass to the system. One command line can consist of up to 1024 characters. (NOTE: By entering a hyphen (–) prior to pressing RETURN, you can enter the command line in segments. Each command line segment can consist of up to 256 characters. The system concatenates the segments into one DCL command prior to interpreting the command.)
Required Elements of a Command Line	
Verb	The verb of the command line is like the verb of a sentence in English. It specifies the action of your request. The verb usually consists of one word. Sometimes, however, it requires a parameter to further specify the action to be taken. This second word is known as an option or a keyword. Example: `$ HELP`
Option or Keyword	The option or keyword is a parameter that may follow the verb. Together with the verb, the option or keyword specifies the action of the instruction. Example: `$ SHOW PROCESS` In the instruction SHOW PROCESS, SHOW is the verb and PROCESS is the option or keyword.
Parameter	The parameter(s) of the command line receive the action of the verb, much like the object of a sentence in English. Example: `$ PRINT FILE1.TXT` In the instruction PRINT FILE1.TXT, PRINT is the verb and FILE1.TXT is the parameter.

Table 2–1 (Cont.) Elements of a DCL Command Line

Command Element	Comments
Optional Elements of a Command Line	
Qualifier	The qualifier(s) of the command line describe or modify the action taken by the verb. Each qualifier is preceded by a slash (/). You can place qualifiers after the verb or after a parameter. Some qualifiers accept one or more values. Format: **/qualifier**[=(**value**[,...])] Example: `$ SHOW PROCESS/ALL` The qualifier /ALL modifies the action. There are three categories of qualifiers. For an explanation of the qualifier types, see Table 2–2.
Value	A value assigns a specific quantity to a qualifier. If you specify more than one value, you must separate the values with commas and enclose them in parentheses. Example: `$ PRINT/COPIES=2 FILE1.TXT` In the above instruction, /COPIES=2 is a qualifier to the verb PRINT. The value of the qualifier is the integer 2.
$	The dollar sign ($) can precede any command. When you are in interactive mode, the DCL CLI ignores the dollar sign. However, the dollar sign must precede commands you place in files. (This technique is discussed in the **Submitting Batch and Print Jobs** module and the **Writing Command Procedures** module.)

Table 2–1 (Cont.) Elements of a DCL Command Line

Command Element	Comments

!	The exclamation mark (!) indicates a comment. The system disregards anything on a command line following an exclamation mark. (The exclamation mark documents commands you place within a file.)
	
	Examples:
	
	`$ !The DCL CLI ignores this comment line`
	
	`$ SHOW PROCESS !This comment is ignored`

Label:	The label is a character string that identifies a particular line in a file that contains DCL commands. Such a file is referred to as a command procedure. You should use labels only in command lines within command procedures.

Table 2–2 The Three Types of DCL Qualifiers

Qualifier Type	Comments

Command	Command qualifiers have the same meaning regardless of where they appear in the command line.
	
	Examples:
	
	`$ PRINT/HOLD FILE1.TXT`
	or
	`$ PRINT FILE1.TXT/HOLD`
	
	Since /HOLD is a command qualifier, the two commands above have the same effect. Both commands place the request in a hold state.

Positional	Positional qualifiers have different meanings depending on where they appear in the command line.
	
	`$ PRINT/COPIES=2 FILE1.TXT,FILE2.TXT`
	
	When you place a positional qualifier after the verb, but before the first parameter, the qualifier affects the entire command line. In this example, the positional qualifier COPIES=2 follows the verb PRINT. Therefore, this command requests the printing of two copies of FILE1.TXT and two copies of FILE2.TXT.
	
	`$ PRINT FILE1.TXT/COPIES=2,FILE2.TXT`
	
	When you place a positional qualifier after a parameter, the qualifier affects only that parameter. In this example, the positional qualifier /COPIES=2 follows the parameter FILE1.TXT. Therefore, this command line requests the printing of two copies of FILE1.TXT and one copy of FILE2.TXT.

Parameter	You can use parameter qualifiers only after a specified type of parameter. There are several types of parameter qualifiers. Refer to the command descriptions in the *VMS DCL Dictionary* for the names and types of all qualifiers that you can use with each command.

2.6.2 DCL Features

In addition to supporting an English–like vocabulary and rules of grammar, DCL supports a number of conveniences. You can:

- Continue DCL commands over more than one line by using the hyphen (–)
- Abbreviate commands and other keywords
- Allow DCL to prompt for required and optional parameters
- Include comments at the end of command lines by using the exclamation point (!)

Table 2–3 describes these special DCL features.

Table 2–3 Features of DCL

Feature	Example	Comments

Continuation	`$ PRINT/COPIES=2 -`
	`_$ FILE1.TXT,FILE2.TXT,-`
	`_$ FILE3.TXT,FILE4.TXT`
		The hyphen continues a command line over more than one line of input.

Abbreviation	`$ LOGOUT ! These are`
	`$ LOGO ! all`
	`$ LO ! equivalent`
		You can always abbreviate commands and keywords to four or fewer characters. Each abbreviation must be unique.

Prompting	`$ PRINT` `_File: FILE1.TXT`	In commands that accept parameters, if you enter the command and press RETURN, DCL prompts for the missing parameters.
Comments	`$ !ALL process information` `$ SHOW PROCESS/ALL`	An exclamation mark precedes comments, used to document commands.

2.6.3 Editing a DCL Command Line

To accomplish various tasks, you will have to perform many of the following operations on a DCL command line:

- Move the cursor
- Issue a command line
- Add or delete data from the command line
- Recall a previously issued command line
- Terminate command execution
- Terminate an operation

Table 2–4 describes how to move the cursor on a DCL command line. Table 2–5 describes how to delete data from a command line. Table 2–6 describes how to add data to a command line and Table 2–7 describes how to recall a previously issued command line.

Table 2–4 Moving the Cursor

Operation	Special Function Key	Comments
Moving the cursor to the left	LEFT ARROW CTRL/D	Moves the cursor one character to the left. Holding the LEFT ARROW key down moves the cursor until the key is released.
Moving the cursor to the right	RIGHT ARROW CTRL/F	Moves the cursor one character to the right. Holding the RIGHT ARROW key down moves the cursor until the key is released.
Moving the cursor to the beginning of the line	CTRL/H	Moves the cursor to the beginning of the line.
Moving the cursor to the end of the line	CTRL/E	The CTRL/E key sequence moves the cursor to the end of the line. This is the cursor's original position.

Table 2–5 Deleting Data from the Command Line

Operation	Special Function Key	Comments
Deleting a character	DELETE	Deletes the character to the left of the cursor. (Note that on a hardcopy terminal the system responds by typing a backslash (\) followed by the DELETE character.)
Deleting a word	LF CTRL/J	Deletes the preceding word.
Deleting a line	CTRL/U	Erases all characters to the left of the cursor. When the cursor is at the end of the line, pressing the CTRL/U key sequence erases the entire command line.
Clearing the line and the type–ahead buffer	CTRL/X	Discards the current line and deletes data in the type–ahead buffer.

Table 2–6 Adding Data to the Command Line

Operation	Special Function Key	Comments
Replacing a character	x	Pressing any keyboard character causes that character to replace the character originally at the cursor position. This is referred to as the Overstrike mode of operation. Overstrike mode is the default data entry mode, equivalent to the following two–step operation: 1. Delete a character 2. Insert a character
Inserting a character	CTRL/A	CTRL/A changes the terminal's data entry mode from Overstrike mode to Insert mode. When you press any keyboard key in Insert mode, the original text moves to the right, making room for the new character(s). If you are in Insert mode, CTRL/A changes the terminal's mode back to Overstrike.

Table 2–7 Recalling a Previously Issued Command Line

Operation	Special Function Key	Comments
Recalling the most recent command line	UP ARROW CTRL/B	Consecutively recalls the last command passed to the DCL CLI. Commands used for recalling previously entered commands from the command buffer are not retained in the command buffer. (Within a program, you can recall only the last command entered.)
Recalling more recently entered commands	DOWN ARROW	Recalls the more recently entered commands from the command buffer. After recalling the *most* recently entered command, pressing the DOWN ARROW key displays a blank line.
Refreshing a DCL command line	CTRL/R	Redisplays the current unentered command line on your terminal. (Note that on a hardcopy terminal, the system issues a RETURN prior to retyping the current command line.)

2.6.3.1 The RECALL Command

The **RECALL** command recalls or displays previously entered commands, so you can re–issue a particular command. Each command issued is automatically stored in the *RECALL* buffer. The *RECALL* buffer stores a maximum of twenty commands. The twenty–first command issued moves the first (Number 1) command out of the *RECALL* buffer; the twenty–second command issued moves the second numbered command out of the *RECALL* buffer, and so on.

The **RECALL** command displays a particular command on the terminal screen, but the system does not process it. Press RETURN to process the command. Table 2–8 describes the various functions of the **RECALL** command.

Table 2–8 Recalling a Previous Command Line with the RECALL Command

Operation	Command/Qualifier	Comments
Displaying a list of the commands you have entered	`RECALL/ALL`	The /ALL qualifier displays a numbered list of the commands you have entered.
Recalling the third entered command line	`RECALL 3`	Adding the parameter 3 to the RECALL command recalls the third entered command line.
Recalling the most recently entered PRINT command	`RECALL PRINT`	The command parameter PRINT recalls the last PRINT command entered.
Erasing the RECALL buffer	`RECALL/ERASE`	The /ERASE qualifier empties the contents of the RECALL buffer.

There are times when you may want to stop and restart information that is displaying on your terminal screen. Table 2–9 describes how to stop and restart the screen display.

Table 2–9 Controlling the Display of Information at your Terminal

Operation	Special Function Key	Comments
Redisplaying the terminal screen	CTRL/W	Within some programs, CTRL/W refreshes the current terminal display. (This key sequence is useful within the default editor (EDT).)
Suspending terminal output	CTRL/S	Suspends the display of information at your terminal by suspending the program that generates it.
	HOLD SCREEN	The HOLD SCREEN key on VT200–series terminals also suspends the display.
	NO SCROLL	The NO SCROLL key on VT100–series terminals also suspends the display.
Resuming terminal output	CTRL/Q	Allows the program suspended by CTRL/S to resume execution. Therefore, it continues the display at your terminal.
	HOLD SCREEN	Depressing the HOLD SCREEN key again after halting the terminal display resumes the display on VT200–series terminals.
	NO SCROLL	Depressing the NO SCROLL key again after halting the terminal display resumes the display on VT100–series terminals.
Suppressing and resuming terminal display	CTRL/O	Suppresses the current image's output. The routine that generates the display continues to execute, returning control to your terminal when it terminates. The command echoes as *Output off*. By entering the key sequence a second time, you enable the terminal to receive output again. The command echoes as *Output off*.

Table 2–10 describes how to cancel a command line and how to close an open file. You may also wish to determine your current process operation by using a control key sequence.

Table 2–10 Terminating an Operation

Operation	Special Function Key	Comments
Canceling a command line	CTRL/Y	Cancels the execution of the current image. Control returns to DCL command level. The command echoes as *Interrupt.*
	CTRL/C	Within certain applications, the CTRL/C key sequence cancels command processing. CTRL/C echoes as *Cancel.* When CTRL/C is not enabled, use CTRL/Y.
Closing a file	CTRL/Z	Indicates the end of a file entered at the terminal. (For example, a file you opened with the CREATE command.) In certain utilities, the CTRL/Z key sequence is equivalent to the EXIT command. (For example, the MAIL utility.) The command echoes as *Exit.*
Determining your current process operation	CTRL/T	Momentarily interrupts output to display a line of statistical information about the current process. This key sequence is only informative. It does not affect the process operation.

2.7 GETTING HELP

As you work on your system, you will find that occasionally you require additional information or advice. To obtain assistance on any given topic, you can consult your system manager or an experienced user. However, there are two means of obtaining self–help:

- A complete documentation set
- An online Help facility

2.7.1 Documentation Set

By using the manuals of the VMS document set, you can locate additional information and examples concerning VMS commands. These manuals also contain definitions of terms and discussions of concepts. Table 2–11 lists the topics for which you may require additional information. The table recommends an appropriate document for each topic.

2.7.2 Online Help Facility

By using the online Help facility, you can obtain information about any VMS command, its parameters, and qualifiers. To invoke the Help facility, type the **HELP** command following the system prompt. The Help facility supplies a list of topics for which information is available and then prompts you for the topics you require. The facility displays topics in either uppercase or lowercase letters. Entering a topic in uppercase yields text on the DCL command of the same name, while entering a topic in lowercase yields text on a general topic. To exit the Help facility, press CTRL/Z. Table 2–12 lists commands for operating the online Help facility.

Table 2–11 Manuals for Locating Information About Your System

Topic	VMS Manual
Commands and qualifiers	VMS DCL Dictionary Refer to each command for a description, syntax, and examples.
Concepts of the operating system	VMS DCL Concepts Manual
Definitions of terms and acronyms	VMS DCL Concepts Manual
Information and error messages issued by the system	VMS System Messages and Recovery Procedures Reference Manual
Location of major topics in the document set	VMS General User's Manual Master Index VMS Master Index VMS Glossary
Restrictions and known problems with current operating system release	VMS Version 5.0 Release Notes

2.8 DOCUMENTATION KITS

There are two sets of documentation available for Version 5.0: the condensed, or Base Set Kit, and the extended set.

2.8.1 Base Set Kit

The *Base Set Kit* includes six manuals.

- *Overview of VMS Documentation*
- *VMS Version 5.0 New Features Manual*
- *VMS General User's Manual*
- *VMS License Management Utility Manual*
- *VMS System Manager's Manual*
  - — Provides an overview of system management concepts and tasks and condensed versions of commonly used system management utilities.
- *VMS Mini–Reference Manual*

The extended set consists of manuals that go into more detail about the system.

You may have either set of documentation at your site, depending on what the system manager orders.

Table 2–12 Using the DCL Help Facility

Operation	Command	Comments
Displaying a list of available help	`$ HELP`	The HELP command lists the topics on which you can obtain information. The system responds with the *Topic?* prompt.
Displaying instructions on HELP	`$ HELP INSTRUCTIONS`	The HELP INSTRUCTIONS command displays detailed instructions on how to use the Help facility.
Displaying a list of hints	`$ HELP HINTS`	The HELP HINTS command produces lists of commands grouped by function.
Displaying information about the SHOW command	`$ HELP SHOW`	The SHOW parameter used with the HELP command produces an informative display on the SHOW command. The system responds with *SHOW Subtopic?* prompt.
Displaying information about the SHOW PROCESS command	`$ HELP SHOW PROCESS`	By including the keyword PROCESS, you can have the system produce an informative display on the SHOW PROCESS command. The system prompt is *SHOW PROCESS Subtopic?*.
Redisplaying the previous HELP screen	?	Pressing the question mark key (?) redisplays the previous Help message.
Returning to DCL command level	RETURN	Moves you one level closer to DCL level. When you are at the *Topic?* prompt, pressing RETURN returns you to DCL command level.
	CTRL/Z	CTRL/Z or EXIT returns you to DCL command level regardless of the HELP prompt. Both commands echo as *EXIT*.

2.9 CHANGING YOUR PASSWORD

The DCL command **SET PASSWORD** changes your password. When changing your password, user input is not echoed at the terminal. You must enter the new password twice. If the two entries do not match, the password does not change.

Example:

```
$ SET PASSWORD

Old password:  QUINOA
New password:  FERMATA
Verification:  FERMATA
```

Note that in a real session, neither the old password nor the new password and its verification would appear on the screen.

2.10 INTERPRETING SYSTEM MESSAGES

The VMS system issues messages in response to the commands you enter at your terminal. These messages tell you if a specified operation was not successful, and suggest possible causes of error. The system does not usually display messages that confirm the success of a command. Figure 2–3 illustrates the format of a system message.

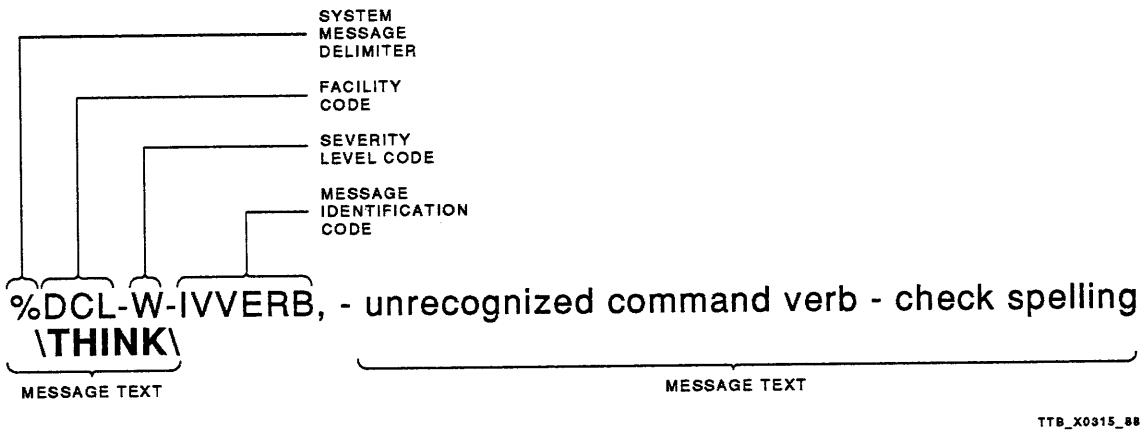


Figure 2–3 The Elements of a System Message

A system message consists of the following parts:

- System message delimiter
- Facility code
- Severity level code
- Message identification code
- Message text

For example, the system message in Figure 2–3 appears in response to the following command:

```
$ THINK
```

This display informs you that the **DCL CLI** does not recognize this verb, and recommends that you check the spelling.

Table 2–13 describes the elements of the system message. Table 2–14 lists the severity levels identified in system error messages.

Table 2–13 Elements of the System Message

Message Element	Example	Comments
System message delimiter	`%, -`	All system messages begin with either a percent sign (%) or a hyphen (–). The percent sign precedes the first system message received, while the hyphen precedes all additional messages.
Facility code	`DCL`	Names the portion of the operating system that detected an error.
Severity level code	`E, F, S, I, W`	Describes the severity of the error. (For an explanation of each severity level code, see Table 2–14).
Message identification code	`IVVERB`	Used to locate further information about a message. (Refer to the *VMS System Messages and Recovery Procedures Reference Manual* for additional information.)
Message text	`unrecognized command verb - check spelling`	Gives a more detailed explanation of the error, and suggests an action to recover from the error.

Table 2–14 Severity Levels in System Error Message

Severity Level	Abbreviation	Comments
Success	S	The VMS system does not usually display success messages.
Informational	I	The VMS system sometimes displays additional information about success of operation.
Warning	W	Some operations may have succeeded, others may have failed.
Error	E	The operation probably failed, but some part may have succeeded.
Severe (or fatal) error	F	The operation failed.

2.10.1 Correcting Errors

To perform various functions to correct errors, you may need to refer to the definitions of various key sequences. Refer to the tables in this module for key sequences related to particular tasks.

For an example of using these key sequences, suppose you type the following command line at your terminal:

```
$ SJOW PPROCESS/ALL
```

The system responds with the message:

```
%DCL-W-IVVERB, unrecognized command verb - check spelling
 \SJOW\
```

The following key sequence recalls, edits, and resubmits the last–entered DCL command line:

1. **CTRL/B** – recalls the last command entered **SJOW PPROCESS/ALL** (the cursor is positioned to the right of "/ALL")
2. **CTRL/H** – moves the cursor to the beginning of the line
3. **CTRL/F** – moves the cursor one character to the right
4. **H** – replaces the "J" with an "H"
5. **CTRL/F CTRL/F CTRL/F CTRL/F** – moves the cursor four characters to the right
6. **DEL** – deletes the character to the left of the cursor
7. **RETURN** – resubmits the entire edited command line to the DCL CLI, regardless of the cursor position

2.11 DISPLAYING CHARACTERISTICS OF YOUR TERMINAL, PROCESS, AND SYSTEM

The characteristics of your terminal, your process, and your system define your working environment. The physical characteristics of your terminal, along with other characteristics your system manager assigns, determine its behavior. You can change some of these settings by using the **SET TERMINAL** command. Before doing so, however, consult your system manager.

A number of system parameters and values stored in your UAF record determine your process characteristics.

Finally, the devices and programs your system supports define its operations. The number of processes currently running and the operations they perform also strongly influence system behavior.

This section shows you how to display information that determines the behavior of your terminal, your process, your system, and your system's network environment. Table 2–15 shows commands for displaying the characteristics of your terminal, process, and system.

2.12 THE SHOW TERMINAL COMMAND

The **SHOW TERMINAL** command displays the current characteristics of your terminal. Each characteristic corresponds to an option of the **SET TERMINAL** command. You would use this command to obtain information regarding the setup of your terminal.

Example:

```
$ SHOW TERMINAL

Terminal: _VTA145:     Device_Type: PRO_Series     Owner: SMITH
Physical terminal: _LTA88:                      Username: SMITH
Input:   9600      LFfill:  0      Width:  80       Parity: None
Output:  9600      CRfill:  0      Page:   24
Terminal Characteristics:
Interactive         Echo                Type_ahead       No Escape
No Hostsync         TTsync              Lowercase        Tab
Wrap                Scope               No Remote        No Eightbit
Broadcast           No Readsync         No Form          Fulldup
No Modem            No Local_echo       No Autobaud      Hangup
No Brdcstmbx        No DMA              No Altypeahd     Set_speed
Line Editing        Overstrike editing  No Fallback      No Dialup
No Secure server    Disconnect          No Pasthru       No Syspassword
No SIXEL Graphics   No Soft Characters  Printer port     Numeric keypad
ANSI_CRT            Regis               No Block_mode    Advanced_video
Edit_mode           DEC_CRT             No DEC_CRT2      No DEC_CRT3
```

2.13 THE SET TERMINAL COMMAND

The **SET TERMINAL** command changes the system's interpretation of the terminal's characteristics. This command allows you to change one or more characteristics of your terminal for a particular application or to override system default characteristics.

The following example changes the width of the terminal screen to 132 characters.

Example:

```
$ SET TERMINAL/WIDTH=132
```

Table 2–15 Commands for Displaying the Characteristics of Your Terminal, Process, and System

Information	Command and Option	Command Qualifier
Time of day	`$ SHOW TIME`	None
Terminal characteristics	`$ SHOW TERMINAL`	None
Space available for your use on your default device	`$ SHOW QUOTA`	None
Process Parameters:		
Default device	`$ SHOW PROCESS`	None
Default directory	`$ SHOW PROCESS`	None
User name	`$ SHOW PROCESS`	None
Priority	`$ SHOW PROCESS`	None
Privileges	`$ SHOW PROCESS`	`/PRIVILEGES`
Process identification code (PID)	`$ SHOW PROCESS`	None
Process quotas and limits	`$ SHOW PROCESS`	`/QUOTAS`
User identification code (UIC)	`$ SHOW PROCESS`	None
Password	None	None

Table 2–15 (Cont.) Commands for Displaying the Characteristics of Your Terminal, Process, and System

Information	Command and Option	Command Qualifier
Names of Processes Running on Your System		
All processes	`$ SHOW SYSTEM`	None
Batch processes	`$ SHOW SYSTEM`	`/BATCH`
Network processes	`$ SHOW SYSTEM`	`/NETWORK`
Subprocesses	`$ SHOW SYSTEM`	`/SUBPROCESS`
Names of all users currently logged in to your system	`$ SHOW USERS`	None
Names of devices on your system	`$ SHOW DEVICES`	None

2.14 SUMMARY

To log in to the system:

- Press RETURN.
- Type your user name and press RETURN.
- Type your password and press RETURN.
  Remember, your password is not displayed.

To log out of the system:

- Type **LOGOUT** and press RETURN.

Table 2–16 DCL Command Line Elements

Command Element	Definition
Command line	A command line is the complete specification of a DCL command.
Required Elements of a Command Line	
Verb	The verb specifies the action of your request.
Parameter	Parameter(s) receives the action of the verb.
Optional Elements of a Command Line	
Qualifier	The qualifier(s) describes or modifies the action taken by the verb. A slash (/) precedes each qualifier.
Value	A value assigns a specific quantity to a qualifier.
$	The dollar sign must precede commands you place in files.
!	The exclamation mark (!) indicates a comment.
Label:	The label is a character string that identifies a particular line in a command procedure.

Getting Help

The documentation set contains information about the VMS system, discussions of concepts, and command definitions and examples.

The online Help facility is invoked by entering the **HELP** command.

System Messages

System messages consist of the system message delimiter, facility code, severity level code, message identification code, and the message text.

2.15 WRITTEN EXERCISE I

Match the letter of a special function key that best describes each of the operations on the following page. You may not use every letter in the list.

Special Function Keys

a. CTRL/B

b. CTRL/O

c. CTRL/Q

d. CTRL/R

e. CTRL/S

f. CTRL/U

g. CTRL/Y

h. DEL or RUBOUT

i. RETURN

Operations

1. _____ You have logged in to your system. A long string of messages, all of which you have seen before, scrolls past on your screen. Suppress the messages, without stopping or aborting the program that produces them.

2. _____ You have just typed the string TYPE FILE&. The cursor is positioned immediately after the ampersand (&). Delete the ampersand.

3. _____ You have entered the SHOW SYSTEM command. A listing of users on your system scrolls past on your screen. Abort further execution of the command and return control to your terminal.

4. _____ You have entered the following string at your terminal:

```
$ SHOW PROCESS/ALL
```

Lines of information scroll past on your terminal screen. Stop the display and halt, but do not abort, the program that generates it.

5. _____ Resume generation of the display that you stopped in the preceding operation.

6. _____ You have made extensive corrections to a command line at a hardcopy terminal. The output looks like this:

```
$ PRYNT \TNY\NT9\9\
  FILIN\NI\
```

Display the line without the echoed corrections.

7. _____ You have just issued a command line. Recall this command.

2.16 LABORATORY EXERCISE I

Complete the following activities at an interactive terminal.

1. Log in to the system, using a legal user name and password.
2. Enter the following command lines at your terminal:
   a. `SHOW TIME`
   b. `SHOW USERS`
   c. `SHOW TERMINAL`
3. Log out of the system.

2.17 LABORATORY EXERCISE II

1. Enter the following command line at your terminal:

   ```
   $ PRODUCE NONESUCH.FIL
   ```

   Since neither the command nor the parameter of the preceding command line exists, the operating system will display one or more error messages at your terminal.

2. Enter the following command line at your terminal:

   ```
   $ TYPE NONESUCH.FIL
   ```

   Since the specified parameter names a file that does not exist, the operating system will display one or more error messages at your terminal.

2.18 LABORATORY EXERCISE III

Log in to your system and use the online Help facility to obtain the information listed below. When you have finished your work, log out.

1. A listing of all topics available through the Help facility
2. A description of the login procedure
3. A description of the **/FULL** qualifier of the **LOGOUT** command
4. A description of the **TIME** option of the **SHOW** command

2.19 LABORATORY EXERCISE IV

1. Log in to your system.
2. Using the **HELP** command, learn as much as you can about the **SHOW** command and its keyword options.
3. Determine the following characteristics of your terminal:
   a. Number of Characters Displayed in an Output Line
   b. Receive Speed
   c. Terminal Type (LA34, VT100, and so on)
   d. Transmit Speed
4. Determine the value of each of the following process parameters:
   a. Account Name
   b. CPU Time Limit
   c. Default Directory Specification
   d. Default Input and Output Device Specification
   e. Priority
   f. Privileges
   g. Process Identification Code
   h. Process Name
   i. User Identification Code
   j. User Name
5. Display the names of all processes running on your system.
6. Display the names of all users on your system.
7. Display the names of all devices on your system.
8. Log out of your system.

2.20 WRITTEN EXERCISE I—SOLUTIONS

Operations

1. b — You have logged in to your system. A long string of messages, all of which you have seen before, scrolls past on your screen. Suppress the messages, without stopping or aborting the program that produces them.

2. h — You have just typed the string TYPE FILE&. The cursor is positioned immediately after the ampersand (&). Delete the ampersand.

3. g — You have entered the SHOW SYSTEM command. A listing of users on your system scrolls past on your screen. Abort further execution of the command and return control to your terminal.

4. e — You have entered the following string at your terminal:

```
$ SHOW PROCESS/ALL
```

Lines of information scroll past on your terminal screen. Stop the display and halt, but do not abort, the program that generates it.

5. c — Resume generation of the display that you stopped in the preceding operation.

6. d — You have made extensive corrections to a command line at a hardcopy terminal. The output looks like this:

```
$ PRYNT \TNY\NT9\9\
  FILIN\NI\
```

Display the line without the echoed corrections.

7. a — You have just issued a command line. Recall this command.

2.21 LABORATORY EXERCISE I—SOLUTIONS

1. Log in to the system, using a legal user name and password.

   - Press RETURN on the terminal keyboard
   - At the *Username:* prompt, type your user name and press RETURN.
   - At the *Password:* prompt, type your password and press RETURN.

   If typed successfully, you should get a "Welcome" message from the system.

   If typed unsuccessfully, the system will output an error message to your terminal, notifying you that either your Username or Password were illegal. Retyping your Username and Password will correct this situation.

2. Enter the following command lines at your terminal:

   a. `SHOW TIME`

   ```
   $ SHOW TIME
   31-DEC-1987  14:10:32
   ```

   b. `SHOW USERS`

   ```
   $ SHOW USERS
             VAX/VMS Interactive Users
              19-DEC-1988 15:35:45.86
       Total number of interactive users = 6
   Username     Process Name      PID     Terminal
   BECKER       BECKER          20200332  VTA115:        LTA76:
   CHAPUT       Mary            20200342  VTA111:        LTA57:
   COVERDALE    COVERDALE       20200390  VTA131:        LTA74:
   GOEHRING     Judy            202003A7  VTA139:        LTA82:
   SMITH        SMITH           20200331  VTA145:        LTA88:
   JOHNSTON     JOHNSTON        20200352  VTA124:        LTA67:
   ```

c. SHOW TERMINAL

Note that the display for your terminal will probably differ in some characteristics from the following example.

```
$ SHOW TERMINAL
Terminal: _RTA2:      Device_Type: VT200_Series  Owner: SMITH
                                                 Username: SMITH
   Input:   9600      LFfill:  0      Width:  80      Parity: None
   Output:  9600      CRfill:  0      Page:   24
Terminal Characteristics:
   Interactive        Echo               Type_ahead          No Escape
   No Hostsync        TTsync             Lowercase           Tab
   Wrap               Scope              No Remote           Eightbit
   Broadcast          No Readsync        No Form             Fulldup
   No Modem           No Local_echo      No Autobaud         Hangup
   No Brdcstmbx       No DMA             No Altypeahd        Set_speed
   Line Editing       Insert editing     No Fallback         No Dialup
   No Secure server   Disconnect         No Pasthru          No Syspassword
   No SIXEL Graphics  Soft Characters    Printer port        Numeric keypad
   ANSI_CRT           No Regis           No Block_mode       Advanced_video
   Edit_mode          DEC_CRT            DEC_CRT2            No DEC_CRT3
```

3. Log out of the system.

```
$ LOGOUT or
$ LOGOUT/FULL
```

2.22 LABORATORY EXERCISE II—SOLUTIONS

1.
```
$ PRODUCE NONESUCH.FIL

%DCL-W-IVVERB, unrecognized command verb - check validity and spelling
 \PRODUCE\
```

The error message displayed at the terminal means that the verb **PRODUCE** is an invalid command. IVVERB is the message code of the error message displayed at your terminal. To correct the error, re–issue the command using a legal DCL command in place of **PRODUCE**.

2.
```
$ TYPE NONESUCH.FIL

%TYPE-W-SEARCHFAIL, error searching for DISK:[SMITH]NONESUCH.FIL;
-RMS-E-FNF, file not found
```

The system notifies you that after searching for the file NONESUCH.FIL on the device DISK: and directory [SMITH] the file cannot be found because it does not exist. To correct the error, re–issue the **TYPE** command followed by a legal file name.

2.23 LABORATORY EXERCISE III—SOLUTIONS

Use the following commands to obtain:

1. A listing of all topics available through the Help facility

```
$ HELP
```

2. A description of the login procedure

```
$ HELP LOGIN
```

3. A description of the **/FULL** qualifier of the **LOGOUT** command

```
$ HELP LOGOUT/FULL
```

4. A description of the **TIME** option of the **SHOW** command

```
$ HELP SHOW TIME
```

2.24 LABORATORY EXERCISE IV—SOLUTIONS

1. To log in to your system, enter the user name and password your system manager assigned to you.

2. To obtain information from the Help utility concerning the **SHOW** command, enter the following command line:

```
$ HELP SHOW
```

   To obtain information about the various optional keywords of the command, enter the preceding command line, followed by a keyword of your choice. For example, to obtain information about the **SHOW PROCESS** command, enter the following command line:

```
$ HELP SHOW PROCESS
```

3. To display the characteristics of your terminal, enter the **SHOW TERMINAL** command.

   a. The WIDTH setting of your terminal determines the number of characters displayed in an output line.

   b. The receive speed of your terminal is the first number specified for the SPEED setting of your terminal.

   c. Your terminal type is the first value that appears following the terminal name in the **SHOW TERMINAL** display.

   d. The transmit speed of your terminal is the second number specified for the SPEED setting of your terminal.

4. To display the specified process parameters, enter the command lines shown below:

   a. `$ SHOW PROCESS/QUOTAS` (Account Name)

   b. `$ SHOW PROCESS/QUOTAS` (CPU Limit)

   c. `$ SHOW PROCESS` (Default Directory Specification)

   d. `$ SHOW TERMINAL` or `$ SHOW PROCESS` (Default Device Specification)

   e. `$ SHOW PROCESS` (Priority)

   f. `$ SHOW PROCESS/PRIVILEGES` (Privileges)

   g. `$ SHOW PROCESS` (PID)

   h. `$ SHOW PROCESS` (Process Name)

   i. `$ SHOW PROCESS` (UIC)

   j. `$ SHOW PROCESS` (User)

5. Display the names of all processes running on your system.

   ```
   $ SHOW SYSTEM
   ```

6. Display the names of all users on your system.

   ```
   $ SHOW USERS
   ```

7. Display the names of all devices on your system.

   ```
   $ SHOW DEVICES
   ```

8. Log out of your system

   `$ LOGOUT` or `LOGOUT/FULL`

CREATING AND EDITING TEXT FILES

3.1 INTRODUCTION

One of the most common tasks on a VMS system is the creation and modification of *text files*. Text files can assume a number of forms and can serve many purposes. They can be:

- Memos and letters
- Data files that are used by other programs and utilities
- Computer programs written in a language like FORTRAN, Pascal, or COBOL

The VMS system provides a number of means for the creation, maintenance, and modification of text files. The two most popular are:

- The *EDT Editor Utility*
- The *Extensible VAX Editor (EVE)*

3.2 OBJECTIVES

To create and modify text files on a VMS system, you should be able to:

- Use the proper DCL command to invoke a text editor:
  - — EDT editor
  - — EVE editor
- Identify the major features of each editor.
- Use appropriate commands and keys to perform editing tasks such as:
  - — Moving the cursor
  - — Adding and deleting text
  - — Selecting and manipulating text strings
- End the editing session.
- Use available online Help facilities.
- Recover files that were being edited at a system interruption.

3.3 RESOURCES

- *Guide to VMS Text Processing*
- *VMS Text Processing Utility Reference Manual*
- *VMS EDT Reference Manual*

3.4 CHOOSING AN EDITOR

There are many reasons for choosing one editor over another. Your choice may be based on ease of editing, the ability to edit more than one file simultaneously, or using multiple buffers and windows.

Restrictions may apply in unique situations, such as having to edit only on a hardcopy terminal.

EDT is the editor supplied with many DIGITAL systems. The *EVE editor* is an editor available only on VMS systems.

Features of both the EDT and EVE editors follow. This listing should aid you in deciding which editor to choose.

3.4.1 EDT Editor Utility

The *EDT Editor* is the default text editing utility supplied with a VMS system. That is, if you enter the **EDIT** command without qualifiers, you invoke the EDT editor.

EDT is available on most DIGITAL systems. It also allows you the ability to edit on a hardcopy terminal. It does, however, impose a bigger system load than EVE.

EDT provides two interactive editing modes: *Line mode* and *Keypad mode.*

3.4.1.1 Line Mode

Line mode editing was originally designed and primarily intended for a hardcopy terminal. It works with a file on a line–by–line basis. Line mode editing, indicated by an asterisk prompt (*), is automatically entered when EDT is invoked.

3.4.1.2 Keypad Mode

Keypad mode requires a video terminal. Keypad mode editing works with the file as a unit, where changes to the screen become changes to the file. This editing mode can be entered by using the **CHANGE** command at the Line mode prompt, or through the invocation of a special initialization file. (Refer to the *Guide to VMS Text Processing* and to the *VMS EDT Reference Manual* for more information on start–up files and the commands they contain.)

3.4.2 The Extensible VAX Editor (EVE)

EVE is primarily a VMS editor. Using EVE, you can create new files as well as manipulate and edit text in these files. You can also edit and manipulate text in existing files.

It does not impose as great a system load as does EDT. It functions only on VT100 and VT200–series and later terminals, and on VAX workstations. However, it cannot be used on hardcopy terminals.

Some of the many features of EVE are:

- Keypad editing
- Insert and Overstrike modes for text entry
- Automatic word wrap
- Multiple windows
- Customization for a user's needs, using the features of the VAXTPU programming language

EVE was designed with both ease of learning and ease of use in mind. Testing has shown it to be easier to learn and use than EDT.

Each editor is discussed in more detail later on in this module. Refer to Appendix A of this module for more information regarding the EDT editor. Refer to Appendix B of this module for more information regarding the EVE editor.

3.5 INVOKING THE EDT EDITOR

Because EDT is the default editor utility, no qualifiers are needed. Note that the command qualifier **/EDT** is available but not required. To invoke the EDT editing utility, enter the command:

$ EDIT file–name

3.5.1 EDT Screen Layout

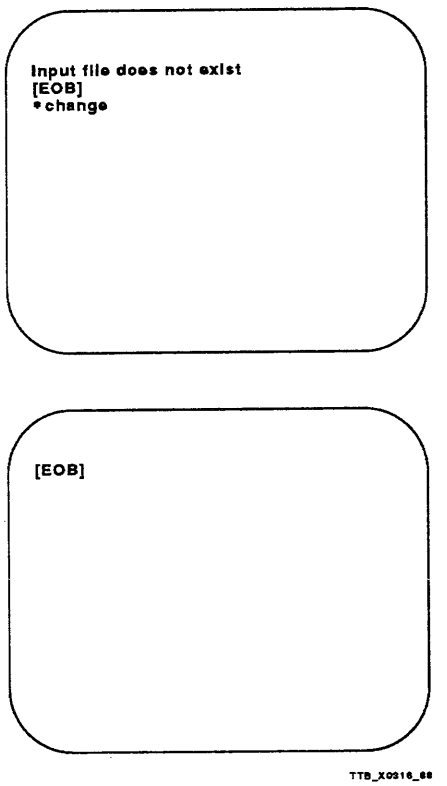


Figure 3–1 EDT Screen Layout – Line Mode and Keypad Mode

Notes on Figure 3–1:

1. The EDT editor is invoked by typing **EDIT file name** at the VMS prompt.
2. When you edit an existing file, the file's contents are copied into a buffer called *MAIN*, and the first line of the file is displayed on the terminal screen.
3. The asterisk (*) prompt indicates you are in Line mode. Enter **CHANGE** to switch from Line mode to Keypad mode. The **CHANGE** command can be abbreviated to the letter **C**.
4. If the text of the file is more than one screen long, the first 22 lines of text appear on the screen.
5. The informational message "Input file does not exist" indicates that you are creating a new file or an existing file name was entered incorrectly.
6. The End of Buffer marker [EOB] indicates the end of the buffer *MAIN*.

3.5.2 Using EDT Help

You can access Help for both Keypad mode and Line mode of the EDT editor.

Line Mode

- Type **HELP** at the asterisk (*) prompt. A brief description of the Help facility is displayed, followed by a list of topics.
- To obtain help on a specific topic, type:

```
*HELP topic-name
```

  The topic name is the name of the command for which you want help. For example, if you want information about the **INSERT** command, type:

```
*HELP INSERT
```

- To return to Keypad mode from Line mode, enter **CHANGE** at the asterisk prompt.

Keypad Mode

- Press PF2 (on VT100–series terminals) or HELP (on VT200–series terminals). A diagram of the keypad functions appears on your terminal screen.
- Press any key on the keypad, or any control key sequence for information on its function.
- To exit from the Help facility, press the space bar. Note that the contents of your file are not affected by using Help.

Example 3–1 describes how to use the Help facility on line. The user first gets a listing of all the topics available and then chooses to obtain help for the **CHANGE** command.

```
$ EDIT MYFILE.TXT
1   Although the computer has always been
*HELP

HELP

You can get help on a topic by typing:

    HELP topic subtopic subsubtopic...

A topic can have one of the following forms:

    1. An alphanumeric string (e.g. a command name, option, etc.)
    2. The match-all or wildcard symbol (*)

Examples: HELP SUBSTITUTE NEXT
          HELP CHANGE SUBCOMMAND
          HELP CH

If a topic is abbreviated, HELP displays the text for all topics that
match the abbreviation.

Additional information available:

CHANGE      CLEAR       COPY        DEFINE      DELETE      EXIT        FILL
FIND        HELP        INCLUDE     INSERT      JOURNAL     KEYPAD      MOVE
PRINT       QUIT        RANGE       REPLACE     RESEQUENCE  SET         SHOW
SUBSTITUTE  TAB         TYPE        WRITE

*HELP CHANGE
```

Example 3–1 Using the Help Facility Online

```
CHANGE

The CHANGE command puts EDT in change mode.  Use change mode to edit at
the character level rather than the line level.

Format: CHANGE [range] [;nokeypad command(s)]

The optional range specifies the cursor position when you enter change
mode.  If you omit range, the current position is used.

There are three submodes of change mode.  Which submode you use depends
on the type of terminal you are using and whether or not you wish to use
the auxiliary (numeric) keypad for editing commands.  These modes are:

    1.  Hardcopy mode
    2.  Keypad mode
    3.  Nokeypad mode

If the CHANGE command contains a semicolon (;) it may be followed by
nokeypad commands.  If the last nokeypad command is EX, EDT returns to
Line mode for the next command line.  This is the only form of the
CHANGE command that may be used in a startup command file or macro.

Additional information available:

ENTITIES   HARDCOPY   KEYPAD     NOKEYPAD   SUBCOMMANDS

*QUIT
```

Example 3–1 (Cont.) Using the Help Facility Online

3.5.3 The EDT Keypad

You can use the defined keys on the EDT keypad to activate alternate definitions for keypad keys using the GOLD key (PF1). You can also move the cursor by character, word, or line; to the top or bottom of files; or in a forward or reverse direction. You can delete and undelete text by character, word, or line. Figure 3–2 shows the screen display of EDT keypad definitions. Table 3–1 describes how to move the EDT cursor. Table 3–2 shows how to change the direction of the cursor. Table 3–3 describes how to delete text in EDT and Table 3–4 shows how to restore text in EDT.

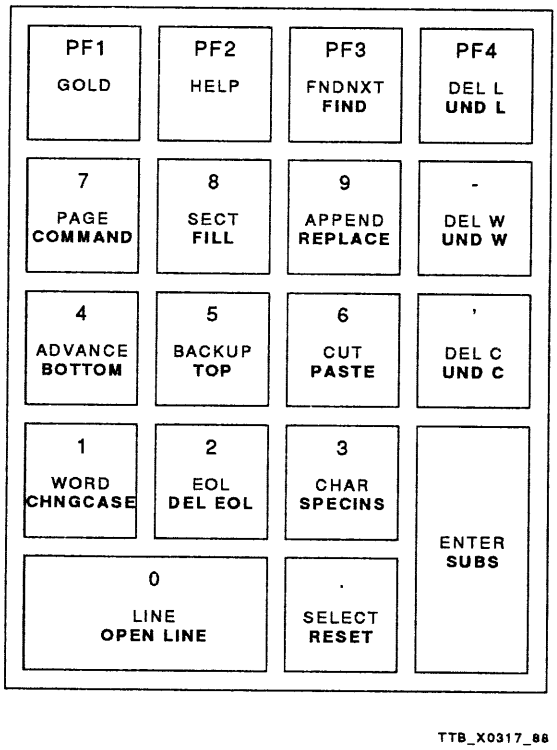


Figure 3–2 EDT Keypad Definitions

Table 3–1 Moving the EDT Cursor

Operation	Key
Move one character in any direction	Arrow keys (UP, DOWN, LEFT, RIGHT)
Move to the beginning of the next line	LINE (0 on keypad)
Move to the beginning of the next word	WORD (1 on keypad)
Move to the end of the line	EOL (2 on keypad)
Move to the next section of the text (16 lines)	SECT (8 on keypad)
Move to the bottom of the buffer	GOLD key (PF1) followed by BOTTOM (4 on keypad)
Move to the top of the buffer	GOLD key (PF1) followed by TOP (5 on keypad)

Table 3–2 Changing the EDT Cursor Direction

Direction	Key
Set cursor to forward	ADVANCE (4 on keypad)
Set cursor to backward	BACKUP (5 on keypad)

Table 3–3 Deleting Text in EDT

Operation	Key	Comments
Delete characters	<X (VT200 series) DELETE (VT100 series)	Deletes the character to the left of the cursor
	DEL C (, on keypad)	Deletes the character on which the cursor is positioned
Delete words	DEL W (– on keypad)	Deletes characters from the cursor position to the beginning of the next word
Delete lines	DEL L (PF4)	Deletes text from the current cursor position to the beginning of the next line
	GOLD/EOL (PF1 followed by 2 on keypad)	Deletes text between the cursor and the end of the line

Table 3–4 Restoring Text in EDT

Function	Keypad Sequence
Restore the last character deleted	GOLD/DEL C (PF1 followed by comma)
Restore the last word deleted	GOLD/DEL W (PF1 followed by hyphen)
Restore the last line deleted	GOLD/DEL L (PF1 followed by key PF4)

3.5.4 EDT File Recovery

The EDT editor provides a journaling facility that keeps track of the changes to the file you are editing. The journaling facility creates a special file, called a *journal file*, that records the keystrokes and editing command used in the editing session.

In the event of a system interruption, you can use this journal file to recover most of the changes you made to the file. An EDT journal file is assigned the same name as the file you are editing and a file type of JOU. (Note that the last few keystrokes or commands may not be recovered.)

To recover your file:

$ EDIT/RECOVER file–name

You must specify the name of the original file (such as MYFILE.TXT), NOT the journal file (MYFILE.JOU). Example 3–2 shows how to recover a file after a system interruption.

```
$ EDIT MYFILE.TXT
*CHANGE
Editing session in progress.
System interruption occurs.
System recovers.
$ EDIT/RECOVER MYFILE.TXT
```

Example 3–2 Recovering a File After a System Interruption

3.5.5 Ending an EDT Session

- Press CTRL/Z to return to the Line mode prompt (*), or
- Press PF1, then keypad key 7. The *Command:* prompt appears at the bottom of the terminal screen.
- At the asterisk, or *Command:* prompt, enter:
  - **EXIT** to exit the session and save your changes. The system creates a new version of the file, and displays an informational message at the bottom of the screen.
  - **QUIT** to exit the session without saving your changes. No changes are made to the file.

3.6 INVOKING THE EVE EDITOR

The *Extensible VAX Editor (EVE)* is an interactive text editor designed for use on video terminals. It has many features that make text editing functions easy to learn and efficient to use. These features include:

- Two editing modes: Keypad mode and Line mode
- Two modes for entering text: Insert mode (text moves to the right as characters are inserted) and Overstrike mode (entered text replaces existing text)
- An online Help facility
- A journaling facility that allows you to recover your work in the event of a system interruption
- Multiple windows that allow you to view the contents of more than one file at the same time
- Automatic word wrap, which automatically begins a new line when the cursor reaches the right margin on the screen

Note that EVE cannot be used on hardcopy terminals.

To invoke EVE, enter the command:

$ EDIT/TPU file–name

If you expect to use the EVE interface frequently, you should create a symbol in your LOGIN.COM file for the command syntax shown above. For example:

```
$ EVE == "EDIT/TPU"
```

3.6.1 EVE Screen Layout

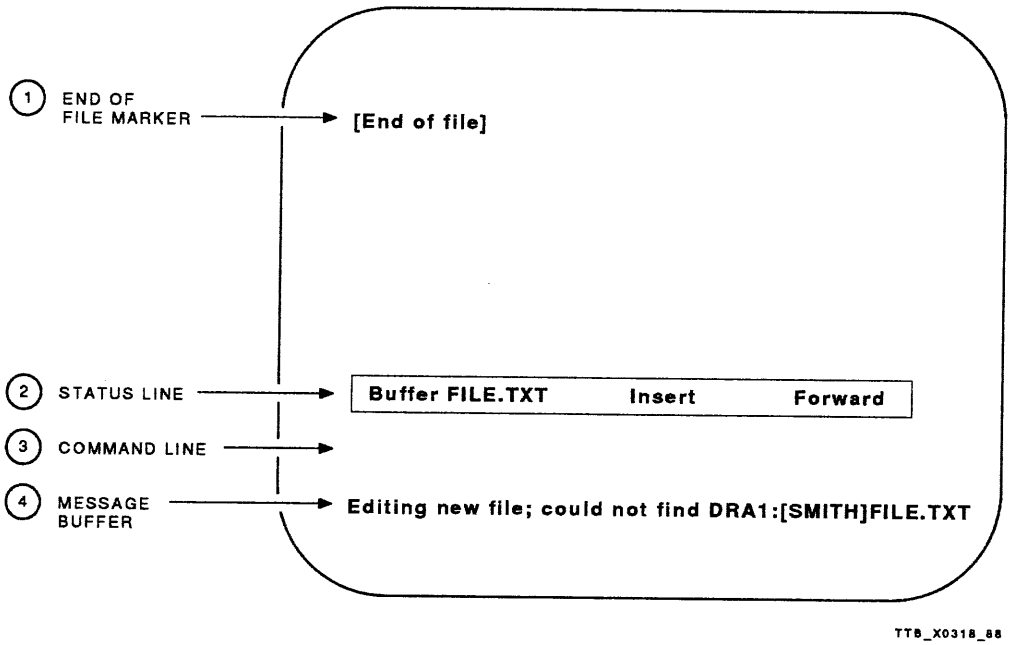


Figure 3–3 EVE Screen Layout

Notes on Figure 3–3:

1 The [End of file] marker marks the end of the file.

2 The status line shows:

The name of the buffer you are currently editing.
The mode for entering text (the mode is Insert).
The direction of the cursor (the direction is Forward).

3 The command line is used to enter Line mode commands. To enter Line mode commands, press DO (PF4 on VT100 terminals) to get the *Command:* prompt.

4 The message buffer stores messages you receive during your editing session.

3.6.2 The EVE Interface

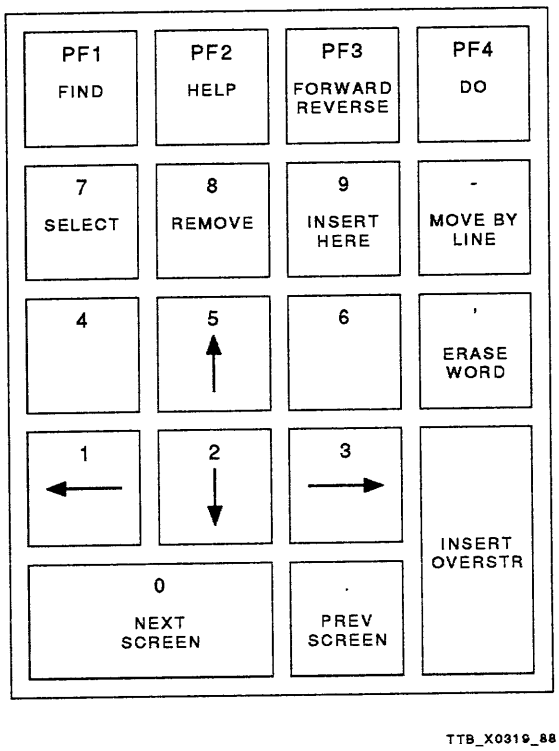


Figure 3–4 EVE Keypad Definitions (VT100–Series Terminals)

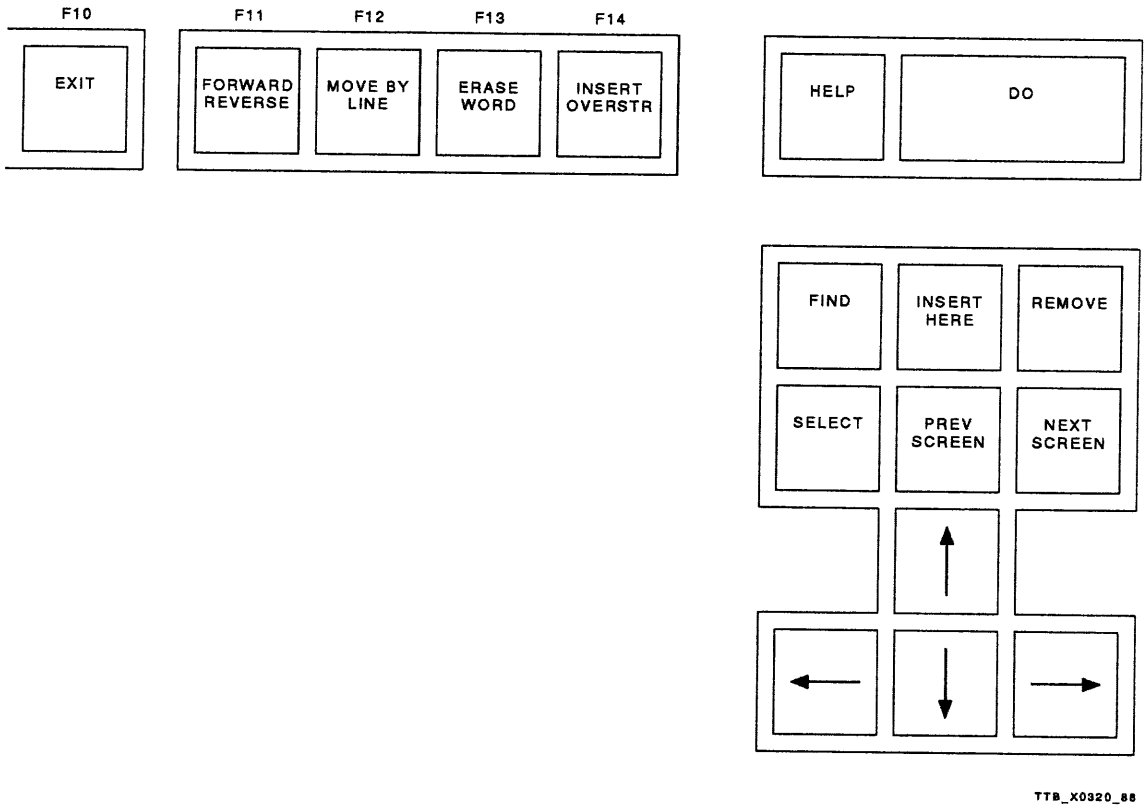


Figure 3–5 EVE Keypad Definitions (VT200–Series Terminals)

3.6.3 Moving the EVE Cursor

The following tables describe the editing keys and EVE commands that move the cursor. Table 3–5 shows how to move the cursor using keys, while Table 3–6 shows how to move the cursor using commands.

Table 3–5 Moving the Cursor Using Keys

Key	Cursor Destination
UP arrow	Moves the cursor up one character.
DOWN arrow	Moves the cursor down one character.
LEFT arrow	Moves the cursor one character to the left.
RIGHT arrow	Moves the cursor one character to the right.
CTRL/E	Moves the cursor to the end of the current line.
CTRL/H	Moves the cursor to the beginning of the current line.
Move By Line	Moves the cursor to the end of the current line or to the end of the next line if the cursor is already at the end of a line, when the current direction is forward. Moves the cursor to the beginning of the current line or to the beginning of the previous line, if the cursor is already at the beginning of a line, when the current direction is reverse.
Prev Screen	Moves the cursor to the previous screen of the current buffer.
Next Screen	Moves the cursor to the next screen of the current buffer.

Table 3–6 Moving the Cursor Using Commands

Command	Cursor Destination
TOP	Moves the cursor to the beginning of the current buffer.
BOTTOM	Moves the cursor to the end of the current buffer.
BUFFER	Puts the specified buffer in the current window, and moves the cursor to the last location it occupied in that buffer. Creates a new buffer if the specified buffer does not exist.
GET FILE [file–name]	Creates a new buffer that contains the text of the specified file, places the new buffer in the current window with the cursor at the beginning of the new buffer. If you specify a nonexistent file, an empty buffer is created.
LINE	Moves the cursor to the beginning of line n in the current buffer (n must be a positive integer).
MOVE BY WORD	Moves the cursor to the beginning of the next word when the current direction is forward. Moves the cursor to the beginning of the previous word when the current direction is reverse.
OTHER WINDOW	When there are two editing windows on your screen, the cursor moves to the last location it occupied in the other editing window.

3.6.4 Inserting Text in EVE

The INSERT/OVERSTR key changes the current editing mode, which is displayed in the status line.

Text is inserted at the current cursor position when in Insert mode, while text already in the file moves to the right.

Text already in the file is overwritten when in Overstrike mode at the current cursor position.

3.6.5 Erasing Text

Table 3–7 Keys for Deleting Text

Key	Effect
DELETE (<X)	Deletes the character to the left of the cursor.
ERASE WORD	Deletes the current word or, if the cursor is not on a word, deletes the next word.
CTRL/U	Deletes all characters from the current cursor position to the beginning of the line.
SELECT	Marks text for removal from the initial cursor position to wherever you move the cursor.
REMOVE	Removes the text that was marked by the SELECT key.

3.6.6 Defining an EDT–Like Keypad

The command **SET KEYPAD EDT** defines an EDT–like keyboard. Note that this command does not enable you to enter EDT commands by using the DO key. You only have access to the EDT–like functions by pressing the keypad keys.

The EDT–style keypad does not fully implement EDT. The differences are:

- CTRL/Z makes EVE write the buffer to a file and exit to the DCL prompt.
- GOLD/KP7 is defined as the DO key when the keypad is set to EDT.
- GOLD/KP8 is defined as FILL, to reformat the currently selected text or the current paragraph. If you want this key to fill only the selected text (as in real EDT) redefine the key as FILL RANGE.
- EVE defines the ENTER key as RETURN.

3.6.6.1 Canceling an EDT–Like Keypad

The command **SET KEYPAD NOEDT** cancels the EDT–like keypad setting.

For VT100–series terminals, this command sets the keypad to VT100, which is the default setting.

For VT200–series terminals, this command sets the keypad to NUMERIC, which is the default setting.

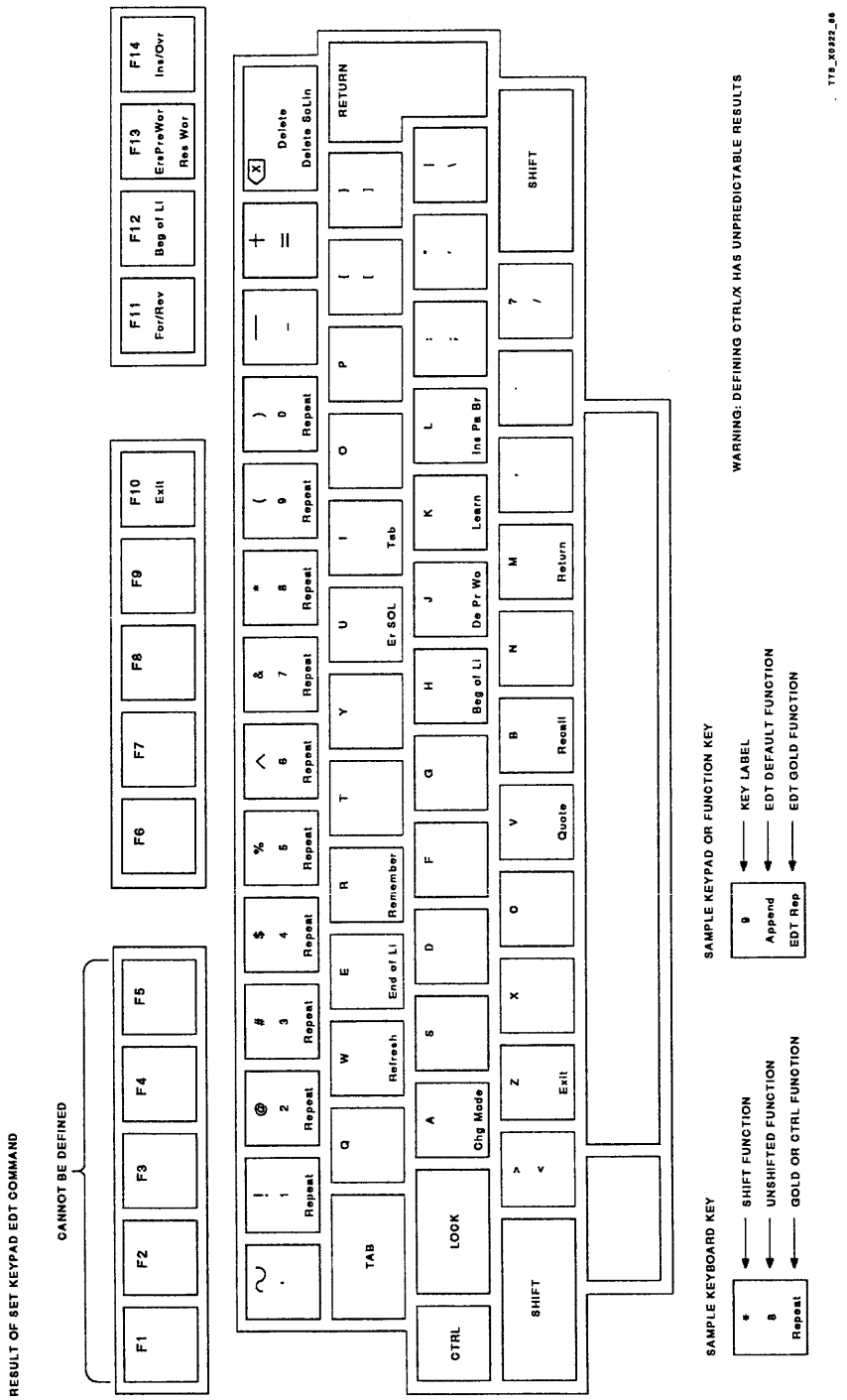


Figure 3–6 EDT–Like Key Definitions for VT200–Series Terminals

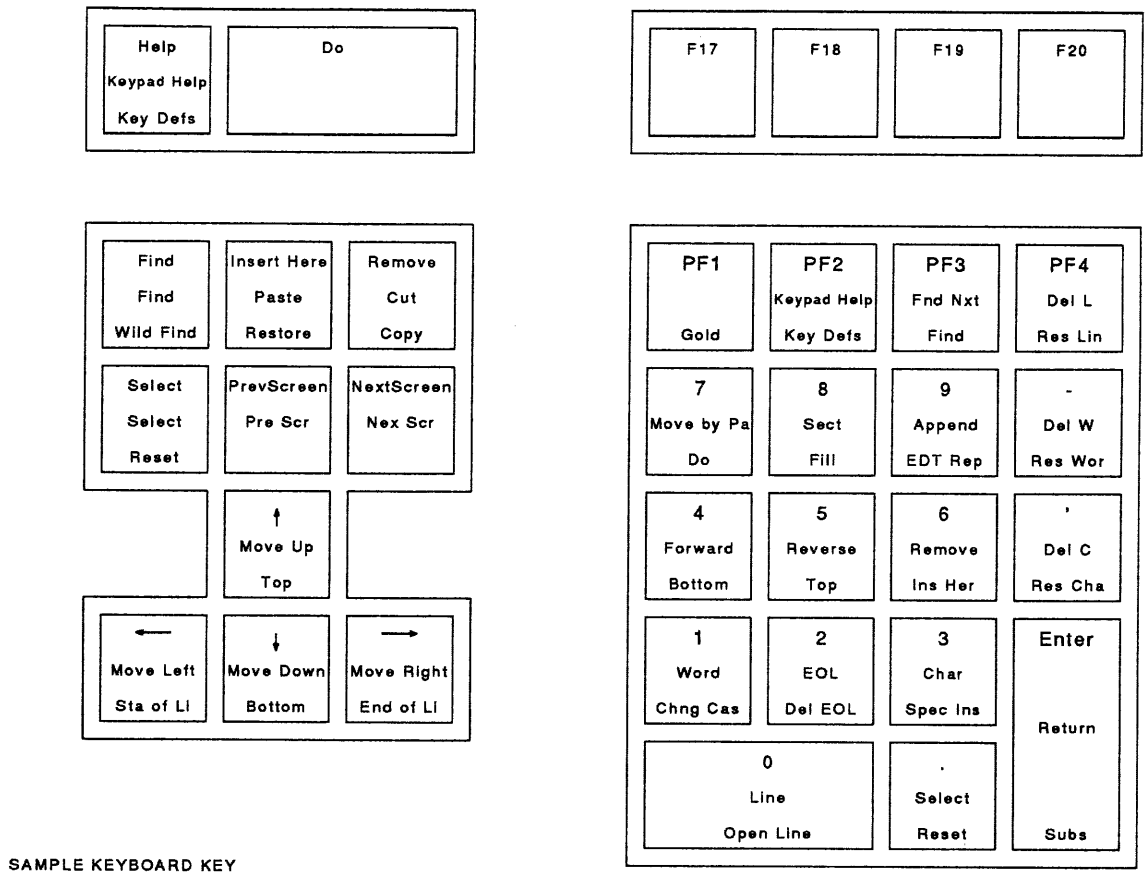


Figure 3–6 (Cont.) EDT–Like Key Definitions for VT200–Series Terminals

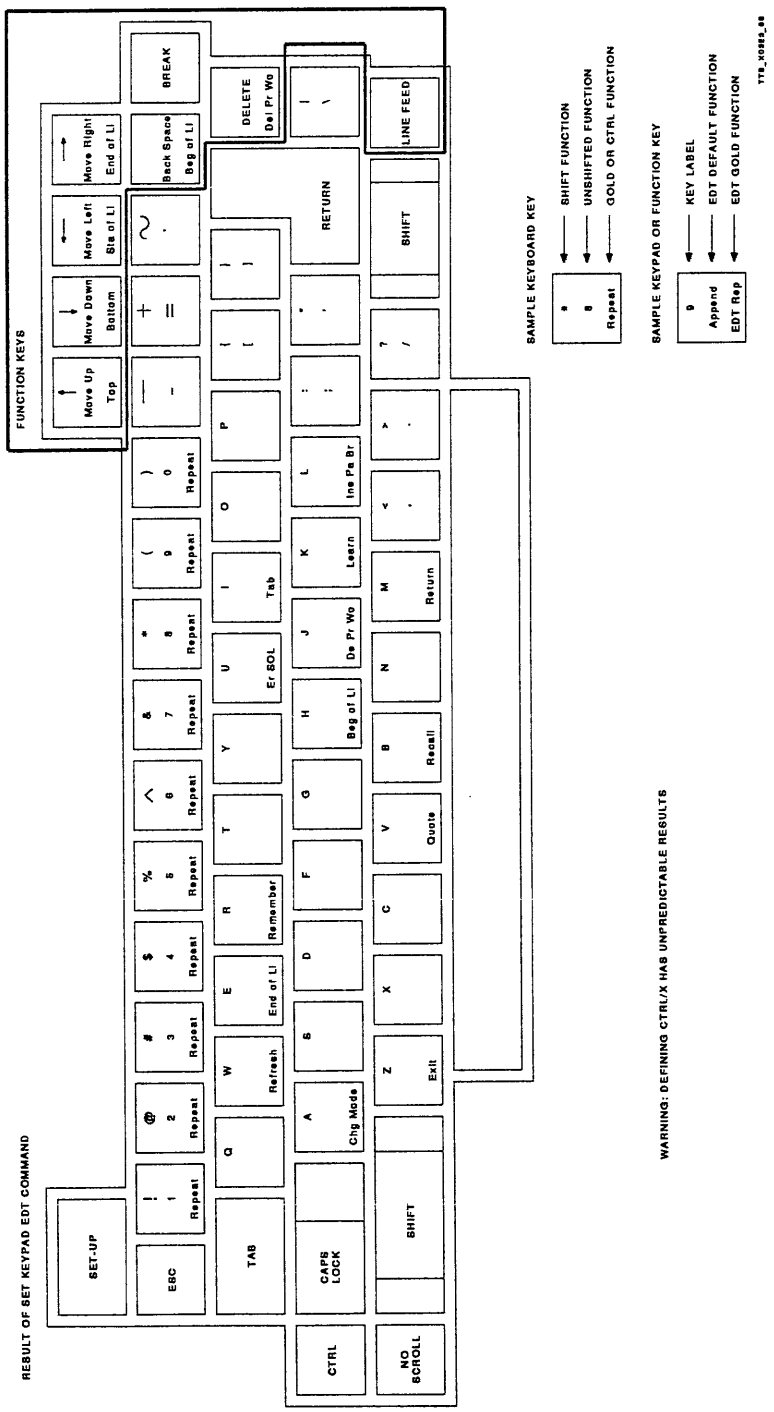


Figure 3–7 EDT–Like Key Definitions for VT100–Series Terminals

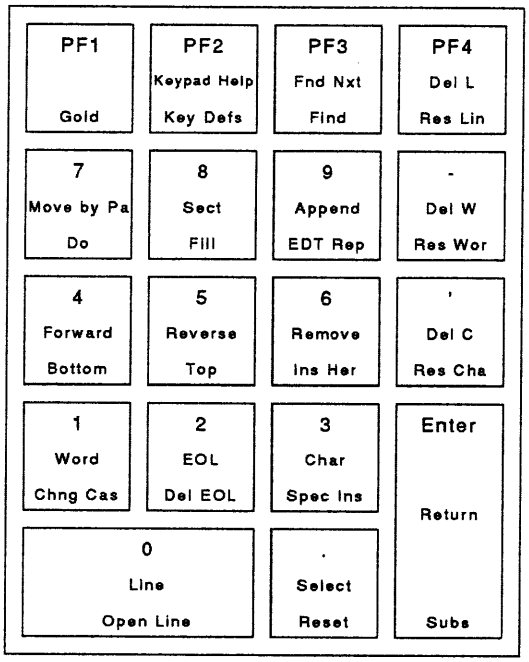


Figure 3–7 (Cont.) EDT–Like Key Definitions for VT100–Series Terminals

3.6.7 Using EVE Help

You can get help on EVE Keypad mode functions and Line mode commands, in the following manner:

Line mode

- Press the DO key (on VT200–series terminals) or PF4 (on VT100–series terminals).
- Type **HELP** and press RETURN.

  A list of commands is displayed on the terminal screen. You can enter any command about which you want information.
- To exit from Help, press RETURN. When you exit from Help, you are placed back in Keypad mode to continue editing your file.

Keypad mode

- Press the HELP key (PF2 on VT100 terminals). A diagram showing the keypad key functions is displayed on the screen.
- Press any key or control key sequence to get information on its function.
- To exit from Help, press RETURN. Note that the contents of your file are not affected by using Help.

3.6.8 File Recovery

Journaling is used to recover files in the event of a system interruption. The journal file default file name is the same as the input file name and the default file type is TJL.

To recover your file:

$ EDIT/TPU/RECOVER file–name

Remember that you must specify the original name of the file, NOT the journal file. Note that the last few changes you made to the file are not recovered. For example:

```
$ EDIT/TPU/RECOVER MYFILE.TEXT
```

3.6.9 Ending an EVE Editing Session

- Press CTRL/Z. This immediately ends the editing session, and saves all of the changes you made during the editing session. If you do not want to save the changes you made to the file:
  - — Press the DO key (PF4 on VT100–series terminals).
  - — At the *Command:* prompt, enter the Line mode command **QUIT**. If you have made modifications, you will be asked if you wish to continue the quitting process. An answer of "Yes" terminates the editing session and returns you to the dollar prompt on the terminal screen, while an answer of "No" allows you to continue editing your file.

3.7 SUMMARY

There are two editors available on a VMS system: EDT (the default editor) and EVE.

Two editing modes are available with EDT:

- Line mode
- Keypad mode

EVE features include:

- Keypad editing
- Insert and Overstrike modes for text entry
- Automatic word wrap
- Multiple windows

The EDT Editor is invoked by:

$ EDIT file–name

The EVE Editor is invoked by:

$ EDIT/TPU file–name

HELP is available within both editors.

An EDT editing session is ended by pressing CTRL/Z then entering **EXIT** or **QUIT**, or by pressing PF1 then keypad key 7 and entering **EXIT** or **QUIT**.

An EVE editing session is ended by pressing CTRL/Z or by pressing either DO or PF4 then entering **EXIT** or **QUIT**.

3.8 APPENDIX A—EDT

3.8.1 Line Mode Editing

EDT's line editing facility can be used with any interactive terminal. Line editing uses the line as its point of reference. Line editing commands are useful for manipulating large blocks of text.

To aid in locating and editing text, EDT assigns line numbers. These line numbers are not part of the text and are not kept in the file when you finish an editing session.

The following sections describe commands used with Line mode editing.

3.8.1.1 Inserting Text

Use the **INSERT** command to insert text. The cursor indents 16 spaces and waits for the text to be inserted. You can enter as many lines as you wish.

Examples:

```
$ EDIT INSERT.FYI
1       This is line 1
*
2       This is line 2
*
3       This is line 3
*
4       etc.
*
[EOB]
*INSERT 2
This is the new text which is being typed
in the file. It is being inserted prior to
line 2
*EXIT

$ EDIT INSERT.FYI
1       This is line 1
*
2       This is the new text which is being typed
*
3       in the file. It is being inserted prior to
*
4       line 2
*
5       This is line 2
*
6       This is line 3
*
7       etc.
*
[EOB]
*QUIT
```

3.8.1.2 Substituting Text

Use either the **SUBSTITUTE** or **SUBSTITUTE NEXT** command to substitute strings of text.

The **SUBSTITUTE** command operates on the current line or on a specified range within the buffer.

Syntax:

***SUBSTITUTE/old–string/new–string**
***SUBSTITUTE/FORTRAN/PASCAL**

To substitute a string throughout the complete buffer, use the **WHOLE** parameter in conjunction with the **SUBSTITUTE** command:

Syntax:

***SUBSTITUTE/FORTRAN/PASCAL/WHOLE**

The **SUBSTITUTE NEXT** command operates on the next occurrence of the specified string within the buffer.

Syntax:

***SUBSTITUTE NEXT/old–string/new–string**
***SUBSTITUTE NEXT/FORTRAN/PASCAL**

3.8.1.3 Moving Text from One Location to Another

Use either the **MOVE** or **COPY** command to move one or more lines of text from one location to another.

Note that the **MOVE** command deletes the text from the original location, whereas the **COPY** command does not delete the text.

Syntax:

***MOVE first range TO second range**

In the following example, lines 20 through 30 are moved above line 10. Note that "second range" always refers to a single line.

```
*MOVE 20 THRU 30 TO 10
```

To move the current line, enter:

```
*MOVE TO 15
```

To move text without deleting the text in the original position, use the **COPY** command.

Syntax:

***COPY first–range TO second–range**

In the following example, lines 35 through 43 are moved above line 7.

```
*COPY 35 THRU 43 TO 7
```

3.8.1.4 Deleting Text

The **DELETE** command deletes lines or group of lines.

Syntax:

***DELETE range**

The following example shows how to delete line 2 in a file.

```
$ EDIT MYFILE.TXT
1   This is Line 1
* TYPE WHOLE
2   This is line 2
3   This is Line 3
4   This is Line 4
* DELETE 2
1 line deleted
3   This is Line 3
* TYPE WHOLE
1   This is Line 1
3   This is Line 3
4   This is Line 4
```

One of EDT's features is the ability to edit more than one file. To accomplish this, you must understand the relationship between buffers and files.

The following section describes the use of buffers in EDT.

3.8.2 Using Buffers in EDT

Buffers are temporary storage areas for text. Buffers enable you to:

- Divide one or more files into sections.
- Move part or all of another file into your editing session.
- Create a file from part or all of the text in a buffer.

When an editing session has started, a buffer called *MAIN* is automatically provided by EDT. The *MAIN* buffer serves as the work area for you.

3.8.2.1 How to Create Buffers

- Press the GOLD key, followed by the COMMAND function.
- At the *Command:* prompt, enter:
  - The **FIND** command
  - An equal sign (=)
  - The buffer name of your choice (buffer names must begin with a letter)

Example:

```
Command: FIND=PASCAL
```

You can now insert and edit text as if you were in the *MAIN* buffer.

To return to the *MAIN* buffer, type:

```
Command: FIND=MAIN
```

3.8.2.2 Copying Text from One Buffer to Another Buffer

The **COPY** command is used to copy the contents of one buffer into another buffer. In the following example, the contents of the buffer *MAIN* are copied into the buffer *FORTRAN*. The current buffer becomes *FORTRAN*.

```
*COPY =MAIN TO =FORTRAN
```

3.8.2.3 Copying Text from a File into a Buffer

The **INCLUDE** command copies a file (outside of EDT) into a buffer. In this example, a file named MYFILE.TXT is copied into the *MAIN* buffer.

```
*INCLUDE MYFILE.TXT =MAIN
```

3.8.2.4 Copying Text from a Buffer Into a File

The **WRITE** command copies text from your current editing session into a file of your choice. In this example, the entire contents of the current buffer are written to a file named MYFILE.FOR.

```
*WRITE MYFILE.FOR
```

3.8.2.5 Deleting Buffers

Use the Line mode command **CLEAR** to delete buffers during an editing session. In the following example, the buffer *PASCAL* is deleted.

```
*CLEAR PASCAL
```

3.9 APPENDIX B—EVE

The following EVE commands are discussed in addition to the commands included in the previous EVE section of the module.

3.9.1 Inserting Text

You can insert:

- Text – Characters will be inserted into the buffer at the current cursor position by typing in the text.
- Files – Files can be inserted using the **INCLUDE FILE** command. The contents of the specified file are inserted into the buffer at the line before the current cursor location.

  Syntax:

  INCLUDE FILE [file–name]
- Special Nonprinting Characters – Press CTRL/V and then press the special nonprinting character.

  Example of inserting a form feed into the buffer:

  Press CTRL/V

  Press CTRL/L

3.9.2 Moving Text from One Location to Another Location

The following keys are used in conjunction with each other to select text, remove it, and then position the text in another location in the current buffer.

SELECT—Marks text for removal from the initial cursor position to wherever you move the cursor.

REMOVE—Removes the text that was marked by the SELECT key and places it in the *INSERT HERE* buffer.

INSERT HERE—Inserts the text in the *INSERT HERE* buffer into the file at the current cursor position.

3.9.3 Locating Text

Syntax:

FIND search–string

The FIND key is used to locate specified text strings.

If the search string is in lowercase characters, EVE disregards the case of letters and locates any occurrence of the string.

If the search string contains one or more uppercase letters, EVE locates only the occurrences of the string that match the search string exactly.

If a search string is found, EVE displays the string in highlighted type. EVE also defines the search string as a select range. While the search string is highlighted, you can perform any operation on it that requires a select range.

3.9.4 Marking Locations in Text

The **MARK** and **GO TO** commands are used when you are editing a large file and want to return to a specific location later on during an editing session.

Press the DO key, enter **MARK label–name**. The label name can be one or more alphanumeric characters.

Press the DO key, enter **GO TO label–name** to move the cursor to the marked location.

3.9.5 Replacing Text

The **REPLACE** command allows you to replace a text string in the current buffer with another text string.

Format:

- Press the DO key
- Enter **REPLACE**
- Enter the string to be replaced following the "Old string" prompt
- Enter the new string at the "New string" prompt

EVE moves the cursor to the first occurrence of the string, and prompts: "Replace? Type yes, no, all, last or quit" The following table lists the response and the action taken by EVE.

Table 3–8 Responding to REPLACE Prompts

Response	EVE's Action
YES	Replaces the string and attempts to locate another occurrence of the string in the current direction.
NO	Does not replace the string, and attempts to locate another occurrence of the string in the current direction.
ALL	Replaces the string and all other occurrences of the string in the current direction. The cursor is moved to the position where the last replacement occurred.
LAST	Replaces this occurrence of the string and stops the REPLACE procedure. The cursor does not move.
QUIT	Does not replace this occurrence of the string and stops the REPLACE procedure. The cursor does not move.

3.9.6 Restoring Text

The **RESTORE** command restores (or undeletes) the last word, sentence, or line erased by any of the following:

- Any EVE command or any key bound to an EVE command
- The DELETE key
- Also includes the PF4, MINUS, KP6, COMMA, and KP2 keys when the keypad is set to EDT

3.9.7 RESTORE CHARACTER

The **RESTORE CHARACTER** command restores (or undeletes) the character last erased by any of the following:

- The **ERASE CHARACTER** command or any key bound to that command
- The DELETE key
- Also includes the COMMA key when the keypad is set to EDT

3.9.8 RESTORE LINE

The **RESTORE LINE** command restores (or undeletes) the line last erased by any of the following:

- The EVE commands **ERASE LINE, ERASE START OF LINE**, or any key bound to these commands
- The GOLD/DELETE key sequence on a VT200–series terminal
- Also includes the PF4 or GOLD/KP2 keys when the keypad is set to EDT

3.9.9 RESTORE WORD

The **RESTORE WORD** command restores (or undeletes) the word last erased by any of the following:

- The **ERASE WORD** command or any key bound to that command
- Also includes the MINUS key when the keypad is set to EDT

3.9.10 Using Buffers in EVE

Buffers are used during an editing session as storage areas. The following table describes the commands that are used to create and manipulate buffers.

Table 3–9 Creating and Manipulating Buffers

Command	Effect
BUFFER	Puts the specified buffer in the current window and moves the cursor to the last location it occupied in that buffer. Creates a new buffer if the specified buffer does not exist.
GET FILE	Creates a new buffer containing the text of the specified file, places the new buffer in the current window, and places the cursor at the beginning of the new buffer. If a file is specified that does not exist, an empty buffer is created.
GO TO	Returns the cursor to the location labeled using the MARK command. If the labeled location is contained in another buffer, EVE moves the cursor to the other buffer and places the buffer in the current window.
SHOW	Displays a screen of information about the current buffer. If more than one buffer is active in the editing session, press the DO key to display information about the other buffers.
WRITE FILE	Writes the contents of the current buffer to a file. If a file name is not specified, EVE uses the buffer name as the file name.

3.9.10.1 Using Multiple Buffers

Use multiple buffers when you want to edit more than one file. This is very useful if you want to move text from one file to another file.

To create a new buffer:

- Press the DO key
- Enter **GET FILE file-name**

To change the buffer in the current window:

- Press the DO key
- Enter **BUFFER buffer-name**

When you exit from using multiple buffers, EVE writes the contents of the current buffer to a file and asks if you want to write the other buffer to a file.

3.9.10.2 Using Multiple Windows

EVE allows you to view multiple windows on your terminal screen at the same time. You can view and edit either two sections of the same buffer (one file) or multiple buffers (multiple files) simultaneously.

The following table lists EVE commands that are used to create and manipulate windows.

Table 3–10 Creating and Manipulating Windows

Command	Effect
TWO WINDOWS	Splits the terminal screen and creates two editing windows, moving the cursor to the last position it occupied in the text of the bottom window.
OTHER WINDOW	Moves the cursor to the last position it occupied in the other window.
ONE WINDOW	Removes the other window from the screen, expanding the current window to occupy the complete screen.
GET FILE	Creates a new buffer containing the text of the specified file, places the new buffer in the current window, and places the cursor at the beginning of the new buffer. If a nonexistent file is specified, an empty buffer is created. After you create two windows on your terminal screen, use the GET FILE command to create a new buffer in one of the windows.
BUFFER	Puts a new buffer in the current window, and moves the cursor to the last position it occupied in the buffer. Creates a new buffer if the specified buffer does not exist. After you create two windows on your terminal screen, use the BUFFER command to put a different buffer in one of the windows.

3.9.10.3 DELETE WINDOW

The **DELETE WINDOW** command deletes the window in which the cursor is located, if you are using more than one window. Be aware that any edits or modifications made to the file in the current window will not be saved.

3.9.10.4 ENLARGE WINDOW

Syntax:

ENLARGE WINDOW integer

This command enlarges the window in which the cursor is located by the number of lines specified. EVE shrinks the other windows on the screen accordingly.

Integer is the number of lines you want to add to the current window. The minimum value is 1. The maximum value is 20 for a VT100 or VT200–series terminal screen.

3.9.10.5 NEXT WINDOW

The **NEXT WINDOW** command moves the cursor from the current window to the window below. If the cursor is already in the bottom window, EVE moves the cursor to the top window.

3.9.10.6 PREVIOUS WINDOW

The **PREVIOUS WINDOW** command moves the cursor from the current window to the window above. If the cursor is already in the top window, EVE moves the cursor to the bottom window.

3.9.10.7 SHRINK WINDOW

Syntax:

SHRINK WINDOW integer

The **SHRINK WINDOW** command reduces the size of the window the cursor is currently in by the number of lines specified. EVE enlarges the other windows accordingly.

Integer is the number of lines by which you want to shrink the window. The minimum value is 1. The maximum value is 9, which is the number of lines by which you can shrink a window if you have only two windows on the screen.

3.9.10.8 SPLIT WINDOW

Syntax:

SPLIT WINDOW integer

The **SPLIT WINDOW** command splits the window in which the cursor is located.

Integer is the number of smaller windows that you want to appear on the terminal screen. If you omit the integer, the current window is replaced with two windows.

3.9.10.9 Editing One File Using Two Windows

To edit two sections of one file at the same time:

- Press the DO key
- Enter **TWO WINDOWS**

Since you are editing one file, any edits that you make in one window are made in the other window, if both windows display the same part of a file. If you are viewing two different sections of the file you may not be able to see the edits being made in the other window.

To move to the other window:

- Press the DO key
- Enter **OTHER WINDOW**

3.9.10.10 Editing Two Files Using Two Windows

To edit two files with two windows:

- Invoke EVE to edit a file
- Press the DO key
- Enter **TWO WINDOWS**
- Enter **GET FILE file–name** at the *Command:* prompt, or
- Enter **BUFFER buffer–name** at the *Command:* prompt

You are now viewing two files on your terminal screen. You can select and remove text from one buffer and insert it into the other buffer by entering **OTHER WINDOW** at the *Command:* prompt.

3.9.11 Defining Keys

You can define keys to execute frequently used EVE commands. You can also save key definitions to be used from one editing session to the next.

EVE does not allow you to define:

- The DO key
- The RETURN key
- The space bar
- All printing characters on the main keyboard

It is recommended that you do not define the following keys and control key sequences:

- DELETE
- F6 (VT200–series terminals)
- HELP (VT200–series), PF2 (VT100–series)
- CTRL/C
- CTRL/R
- CTRL/S
- CTRL/T
- CTRL/U
- CTRL/Q
- CTRL/X
- CTRL/Y

To define a key:

- Press the DO key
- Enter **DEFINE KEY**
- Enter the key to be associated with the EVE command

A message "Key defined" appears if you have successfully defined a key.

3.9.11.1 Saving Key Definitions

The **SAVE EXTENDED TPU** command saves all key definitions in a section file that you specify. This command must be executed before ending an editing session.

Syntax:

SAVE EXTENDED TPU device:[directory]file–name.TPU$SECTION

You can include the device, directory, and file name that you choose. However, the section MUST be TPU$SECTION.

If you specify the same file name each time you execute the **SAVE EXTENDED TPU** command, all key definitions will accumulate in the same file from all editing sessions.

3.9.11.2 Using Key Definitions

To use this extended version of EVE, you must include the **/SECTION** qualifier when invoking EVE.

Syntax:

EDIT/TPU/SECTION=device:[directory]file–name.TPU$SECTION file–name

Example:

```
$ EDIT/TPU/SECTION=DISK:[SMITH]EVEDEFS.TPU$SECTION MYFILE.TXT
```

3.10 INTRODUCTION TO THE LABORATORY EXERCISES

You should feel free to choose either the EDT editor exercises or the EVE editor exercises. Choose the editor you will be primarily using.

However, if you would like practice in both editors, you can complete all the exercises for this module.

3.11 LABORATORY EXERCISE I (THE EDT EDITOR)

1. Use the EDT editor to create a text file named EXERCISE1.TEXT.
   a. Invoke the EDT editor, using the appropriate DCL command.
   b. Notice the message displayed on the terminal screen.
   c. Change from Line mode to Keypad mode.
2. Before you begin entering text, you should become familiar with the Help facility that is a part of the EDT editor.
   a. Invoke the Help facility from Keypad mode by pressing the appropriate key.
   b. Display information about specific keypad keys.
   c. Exit from Help.
3. Type in the following text:

```
The purpose of this exercise is to allow
you practice using the basic capabilities
of the EDT Editor Utility.
```

4. End the editing session normally.
   a. Return to Line mode.
   b. Type in the command that ends the session and saves your actions.
   c. Notice the system message displayed on the terminal screen.

5. Begin another editing session using the file EXERCISE1.TEXT.
   a. Invoke the EDT Editor, using the appropriate DCL command.
   b. Notice that the first line of the file is displayed on the terminal screen.
   c. Change from Line mode to Keypad mode. The file's contents are displayed on the screen.
6. Modify the text.
   a. Using the appropriate keys, move the cursor to the beginning of the word "basic" in the second line.
   b. Delete the words "basic capabilities" and modify the line so that it reads:

```
you practice using the simpler functions
```

7. End the editing session.

3.12 LABORATORY EXERCISE II (THE EVE EDITOR)

1. Use the EVE editor to create a file called EXERCISE3.TEXT.

   a. Notice the messages that are displayed on the terminal screen.

2. Type in the lines listed below. DO NOT press RETURN while you are typing. The automatic word wrap feature causes new lines to begin when text reaches the right margin of the terminal screen.

```
This is an exercise that uses the EVE editor.
This editor allows you to type text into a file.
The word wrap feature will automatically wrap lines as you type,
so that you do not have to press the RETURN key at the end of each line.
```

3. End the editing session and save your work.

   a. Use the appropriate key sequence or line–mode command to end the editing session.

   b. Notice the system messages displayed on the terminal screen.

4. Begin another editing session using the same file.

   a. Notice the system messages displayed on the terminal screen.

5. Modify the text of the first line to read:

```
This is an example that uses the EVE editor.
```

   a. Move the cursor to the beginning of the word "exercise."

   b. Use the appropriate key to delete the word "exercise".

   c. Insert the word "example" before the word "that".

6. End the editing session normally.

3.13 LABORATORY EXERCISE III (THE EVE EDITOR)

This exercise lets you practice editing more than one file on your terminal screen.

1. Edit a file of your choice.
2. Split your terminal screen into two windows.
3. Edit another file of your choice.
4. Move text from one file into the other file.
5. Exit the file so that the moved text is saved.

3.14 LABORATORY EXERCISE I (THE EDT EDITOR)—SOLUTIONS

1. To create the file:

   a. Enter the command:

   ```
   $ EDIT EXERCISE1.TEXT
   ```

   b. The system displays the following:

   ```
   Input file does not exist
   [EOB]
   *
   ```

   The message indicates that the file EXERCISE1.TEXT did not previously exist in your directory. The [EOB] marker indicates the end of the buffer. The asterisk indicates that you are in Line mode.

   c. To enter Keypad mode, enter the **CHANGE** command at the Line mode prompt:

   ```
   *CHANGE
   ```

   The screen display erases, and the [EOB] marker appears in the upper left corner of the screen.

2. To become familiar with the available Help:

   a. Invoke the Help facility from Keypad mode by pressing the appropriate key:

   Press the PF2 key on the keypad.

   or

   Press the HELP key (on the VT200 keyboard).

   The keypad diagram is displayed on your terminal screen.

b. You can display an explanation of each defined key by pressing the key while in Help. You might want to begin with the following keys:

- HELP (PF2)
- DELETE
- DOWN ARROW
- GOLD (PF1)
- DELETE/UNDELETE LINE (PF4)

Examine any of the key definitions you wish.

c. To exit Help:

Press the space bar

3. Type in the text as indicated.

Note that EDT does not automatically wrap at the end of a line. You must explicitly press RETURN to insert carriage returns into the text.

4. To end the editing session and save your work:

a. Press CTRL/Z. This returns you to Line mode.

b. At the Line mode prompt, enter the **EXIT** command.

```
*EXIT
```

c. The system displays the full file specification and the file's length in lines. The system then returns you to the DCL level.

```
DISK:[SMITH]EXERCISE1.TEXT;1 3 lines
$
```

5. To edit the file:

   a. Enter the command:

```
$ EDIT EXERCISE1.TEXT
```

   b. The system displays the following:

```
1 The purpose of this exercise is to allow
*
```

   The first line of text is displayed on the screen.

   c. At the Line mode prompt, enter **C**. The contents of the file EXERCISE1.TEXT are displayed on the screen.

6. To modify the text:

   a. Move the cursor to the beginning of the word "basic." You can do this by using either the arrow keys, or the keypad keys 0 (zero) and 1 (one). The keypad key 0 moves the cursor from line to line; the keypad key 1 moves the cursor from word to word.

   b. To delete a word, press the HYPHEN (–) key (below PF4 on the keypad). The word to the right of the cursor is deleted. If you press it again, the next word to the right of the cursor is deleted.

   Type in the modifications to the text as shown.

7. End the editing session normally by pressing CTRL/Z, and then entering **EXIT** at the Line mode prompt.

3.15 LABORATORY EXERCISE II (THE EVE EDITOR)—SOLUTIONS

1. To create the file, enter the command:

```
$ EDIT/TPU EXERCISE3.TEXT
```

   a. You should see messages at the bottom of the screen indicating that the file EXERCISE3.TEXT did not previously exist in your directory. The cursor should be positioned at the top of the screen, next to the end of file marker.

      The status line appears at the bottom of the screen. It contains the name of the buffer. In this case, the buffer name is the same as the file name. In addition, the status line indicates the editing mode and search direction. The default values for these are Insert and Forward.

2. Type in the text as indicated.

   Note that the EVE editor automatically wraps at the end of a line. You need not press RETURN to insert carriage returns into the text.

3. To end the editing session and save your work:

   a. Press CTRL/Z. This automatically ends the editing session.

      You can also end an EVE editing session using a Line mode command.

      Press PF4 or DO. At the *Command:* prompt, enter the **EXIT** command.

   b. An informational message is displayed that includes the full file specification and the number of lines in the file. The system then returns you to the DCL level.

      NOTE
      If you wish to end an EVE editing session without saving changes, you must exit using Line mode. Press PF4 or DO, and at the prompt, enter QUIT. You will be asked if you wish to continue the quitting process. Enter Y.

4. Enter the command:

```
$ EDIT/TPU EXERCISE3.TEXT
```

   a. A message is displayed indicating that four lines were read from the file, and the contents of the file appear on the terminal screen. The cursor is at the top of the file.

5. To modify the file:

   a. Use the arrow keys to move the cursor to the beginning of the word "exercise."

   b. There are two ways to delete words, depending on the terminal you are using:

      - VT100—Press the COMMA (,) key on the keypad. The word to the right of the cursor is deleted.
      - VT200—Press the F13 key along the top of the keyboard. The word to the right of the cursor is deleted.

   c. Type in the word "example".

6. End the editing session normally by pressing CTRL/Z.

3.16 LABORATORY EXERCISE III (THE EVE EDITOR)—SOLUTIONS

This exercise lets you practice editing more than one file on your terminal screen.

1. Edit a file of your choice.

```
$ EDIT/TPU FILE1.TXT
```

2. Split your terminal screen into two windows.
   Press DO or PF4 and, at the command prompt, enter:

```
Command: TWO WINDOWS
```

3. Edit another file of your choice.

```
Command: GET FILE FILE2.TXT or  GET FILE2.TXT
```

4. Move text from one file into the other file.

   It is your choice as to which commands you use to move text.

5. Exit the file so the moved text is saved.

```
Command: EXIT
```

COMMUNICATING WITH OTHER USERS

4.1 INTRODUCTION

The ability to communicate with users, both on your system and on other VMS systems is invaluable. You may wish to confirm meetings, help in solving problems, ask a system operator to mount a magnetic tape, etc. To facilitate communications, there are two VMS utilities and one DCL command you can use. They are:

- The Mail utility (MAIL), which allows you to send messages to other users, both on your system and on other systems.
- The Phone utility (PHONE) for communicating interactively with other logged–in users both on your system and on other systems.
- The **REQUEST** command, which allows you to send a message to a system operator's terminal and, optionally, requests a reply.

4.2 OBJECTIVES

To communicate with other users, you should be able to:

- Use the Mail utility to:
  - — Send and receive mail messages.
  - — Print and delete messages.
  - — Organize messages by using Mail folders.
- Place and answer calls using the Phone utility.
- Send messages to a system operator using the **REQUEST** command and optionally receive a reply from the operator.

4.3 RESOURCES

- *VMS Mail Utility Manual*
- *VMS Phone Utility Manual*
- *VMS DCL Dictionary*

4.4 THE HELP FEATURE OF VMS UTILITIES

Most of the VMS utilities contain a Help feature. This online Help feature is invaluable when, for example, you want to use a particular command but do not remember the correct syntax. You can use this Help feature any time you are within a VMS utility. Whenever you exit from Help, you automatically return to your previous position in the utility. In the following steps, the Mail utility is used as an example of using the Help feature.

To use a utility's Help feature:

- Enter the command that invokes the utility. This command is usually the name of the utility, such as MAIL or PHONE. The screen prompt that appears on the terminal screen is the name of the utility.
- Enter the **HELP** command at the utility's prompt:

  ```
  MAIL> HELP
  ```

- Follow the online instructions.
- To exit the utility, enter the **EXIT** command following the utility's prompt or CTRL/Z, which brings the utility's prompt to the screen, followed by another CTRL/Z.

4.5 THE MAIL UTILITY

Using the Mail utility, you can send messages to any user who has an account on your system or to other users on other VMS systems. You can also read, file, forward, delete, print, and reply to messages that other users send to you.

4.5.1 Organization of Mail Messages

By default, a file named MAIL.MAI stores Mail messages. The system automatically creates this file for you.

Mail catalogues messages in folders. Three of these folders are named:

- *NEWMAIL* – Contains new messages you have not yet read.
- *MAIL* – Contains old messages you have already looked at.
- *WASTEBASKET* – Contains messages marked for deletion.

The *WASTEBASKET* folder is emptied automatically when you exit from Mail.

4.5.2 Using the Mail Utility

The Mail utility is invoked by entering the name of the utility at the DCL prompt.

Example:

```
$ MAIL
MAIL>
```

You can now enter commands to:

- Read and send messages to other users
- Obtain a list of messages you have received
- Delete old messages
- Exit from the Mail utility

4.5.3 Reading a Message

When you log in to the system, you are notified if you have received any messages since the last time you logged out of the system.

The Mail utility displays an informational message on your terminal screen when any messages are sent, if you are currently logged in.

The **READ** command is used to read any messages you have received. Pressing RETURN is the same as entering the **READ** command. If a new message arrives while you are in the Mail utility, issue the command **READ/NEW** to read the message.

- To read the first new message:
  - — Invoke the Mail utility.
  - — Press RETURN at the *MAIL>* prompt, or
  - — Enter **READ** at the *MAIL>* prompt
- To read a message you receive while within the Mail utility:
  - — Enter the **READ/NEW** command:

```
MAIL> READ/NEW
```

```
$ mail
You have 1 new message.
MAIL>
#1          12-DEC-1987 09:19:25                          NEWMAIL
From:     SPEEDY::JIM
To:       SMITH
Subj:     Status meeting

John,

I will be out of town so I will not be able to attend the status
meeting.  Fill me in when I get back.

Jim

MAIL>
```

Example 4–1 Reading a Mail Message

Table 4–1 describes various commands used to read a mail message.

Table 4–1 Mail Commands Used to Read a Message

Operation	Example	Comments
Displaying the contents of a message in the current folder of the current file	`READ n`	Displays the message associated with the message number (n)
	`READ`	Displays the next page of the current message having the next–highest message number
	`READ/NEW`	Displays new messages that arrived while you are in the Mail utility
	`NEXT`	Displays the first page of the message having the next–highest message number
	`FIRST`	Displays the contents of the message having the lowest message number
	`LAST`	Displays the contents of the message having the highest message number
	`CURRENT`	Displays the contents of the current (last–read) message
	`BACK`	Displays the contents of the message preceding the current message

4.5.4 Sending a Message

The **SEND** command sends a message to other users. The Mail utility prompts you first for the name of the user(s). It then prompts you for the subject. Enter the message you wish to send, then press CTRL/Z. Note that once you have typed a line and pressed RETURN, you cannot go back and edit that line again.

If you decide not to send a message, but wish to stay in the Mail utility, press CTRL/C to abort the message. Pressing CTRL/Y exits you from the Mail utility.

You can include a file specification with the **SEND** command. If this is done, the text of that file is sent to the specified user(s). For example:

```
MAIL> SEND MYFILE.TXT
To: USER    (or)    USER1,USER2
Subj: Meeting next week
```

The text within MYFILE.TXT is now sent to the user(s)

```
MAIL> EXIT
```

Example 4–2 shows how to send a message to a user.

```
$ MAIL

MAIL> SEND
To:      SMITH
Subj:    Department Meeting
Enter your message below. Press CTRL/Z when complete, or CTRL/C to quit:

Jim,

There will be a department meeting on Friday at 9:00 am.
Please attend if at all possible.
John
CTRL/Z
MAIL>
```

Example 4–2 Sending a Mail Message

Table 4–2 Mail Utility Commands Used to Send Messages

Operation	Format/Example	Comments
Routing a message or the contents of a file to a user or group of users	**MAIL> SEND [/qualifier] [file–specification]**	

	`MAIL> SEND`
	`To: JONES,ALAN`
	`Subj: Today's Agenda`
		Sends the contents of your message to each user listed after the *To:* prompt. You enter the message without editing capabilities.

	`MAIL> SEND/EDIT`
	`To: @DISTRIBUTION.DIS`
	`Subj: Today's Agenda`
		Sends the contents of your message to each user listed in the file named DISTRIBUTION.DIS. You enter the message by using the EDT editor. The message is sent when you exit the editor.

	`MAIL> MAIL MYFILE.LIS`
	`To: JONES`
	`Subj: Today's Agenda`
		Sends the contents of the file MYFILE.LIS to the mail file of the user JONES.

Routing a copy of the current (last–read) message to a user or group of users	`MAIL> FORWARD`
	`To: JONES`
	`Subj: Good News!`
		Sends a copy of the current message to each user listed after the *To:* prompt.

Routing a message or the contents of a file to the sender of the current (last–read) message	**ANSWER** **REPLY [/qualifier] [file–specification]**	
	`MAIL> REPLY` `Subj: You're Right!`	The REPLY or REPLY/EDIT command sends your message to the sender of the current message. (REPLY and ANSWER are synonyms.)

4.5.5 Displaying a List of Messages

The **DIRECTORY** command displays a numbered list of your messages, including the message number, sender's name, date, and subject. To read an old message, enter the **DIRECTORY** command, and after obtaining the list of messages, enter the desired message number.

Example 4–3 describes how to obtain a listing of messages and how to read message number 3.

```
MAIL> DIRECTORY

                                                                    MAIL
# From                    Date          Subject

1 SPEEDY::JIM             12-DEC-1987   Status meeting
2 SPEEDY::SMITH           12-DEC-1987   Schedule of meetings
3 SPEEDY::JONES           12-DEC-1987   Party

MAIL> 3

#3          12-DEC-1987 09:22:53                                    MAIL
From:     SPEEDY::JONES
To:       SMITH
Subj:     Party

John,

We're having an office party next Thursday.
Would you like to come?

Tom

MAIL>
```

Example 4–3 Listing and Reading Old Messages

4.5.6 Deleting a Message

The **DELETE** command marks a message for deletion. Either a single message or a range of messages can be deleted. If a message number is omitted, this command marks the message you are currently reading for deletion.

The **DELETE** command moves all messages marked for deletion to the *WASTEBASKET* folder. When you exit from the utility, the *WASTEBASKET* folder empties automatically.

Deleted messages can be recovered from the *WASTEBASKET* folder by using the **MOVE** command.

When you issue the **DIRECTORY** command after deleting a message, the directory list displays the message number(s) that has been marked for deletion.

Here are some sample DELETE commands:

```
MAIL> DELETE message-number[,...]

MAIL> DELETE 3 (Deletes message number 3)

MAIL> DELETE 1,3,5-7 (Deletes message numbers 1, 3, 5, 6, 7)
```

Example 4–4 shows what happens when you delete a message.

```
MAIL> DELETE 3

MAIL> DIRECTORY
MAIL
#  From                 Date          Subject

1  SPEEDY::JIM           12-DEC-1987  Status meeting
2  SPEEDY::SMITH         12-DEC-1987  Schedule of meetings
3  (Deleted)
```

Example 4–4 Deleting a Mail Message

4.5.7 Getting Help on Mail Utility Commands

Use the **HELP** command at the *MAIL>* prompt to obtain information about the Mail utility. Example 4–5 demonstrates how to enter the **HELP** command and then obtain help on the Mail command **READ**.

```
MAIL> HELP

HELP

    Allows you to obtain information about the Mail Utility.

    To obtain information about all of the Mail commands, enter the
    following command:

         MAIL> HELP *

    To obtain information about individual commands or topics, enter
    HELP followed by the command or topic name.

     Format:

         HELP [topic]

  Additional information available:
  /EDIT       /PERSONAL_NAME        /SELF       /SUBJECT    ANSWER      ATTACH
  BACK        COMPRESS    COPY      CURRENT     DEFINE      DELETE      DIRECTORY
  EDIT        ERASE       EXIT      EXTRACT     FILE        FIRST       Folders
  FORWARD     GETTING_STARTED       HELP        KEYPAD      LAST        MAIL
  MARK        MOVE        NEXT      PRINT       PURGE       QUIT        READ
  REMOVE      REPLY       SEARCH    SELECT      SEND        SET-SHOW    SPAWN
  V5_CHANGES

Topic? READ
```

Example 4–5 Getting Help for Mail Utility Commands

```
READ

    Displays your messages.  It can be issued with or without
    parameters.

    Pressing the RETURN key is the same as entering the READ command
    without parameters.  If you issue the READ command without
    parameters or press RETURN immediately after MAIL is invoked, MAIL
    displays the first page of your oldest unread message in your
    NEWMAIL folder.  If there are no unread messages, MAIL displays the
    oldest message in the MAIL folder.  Each time you enter the READ
    command without parameters, or press RETURN, MAIL displays the next
    page, or the next message if there are no more pages in the current
    message.

    If a new message arrives while you are in MAIL, you can enter
    READ/NEW to read the message, and then return to the previous MAIL
    activity.

     Format:

         READ [folder-name] [message-number]

  Additional information available:

  Parameters Qualifiers
  /BEFORE    /CC_SUBSTRING          /EDIT      /FROM_SUBSTRING       /MARKED
  /NEW       /REPLIED   /SINCE      /SUBJECT_SUBSTRING    /TO_SUBSTRING
  Examples

READ Subtopic?
Topic?

MAIL> EXIT
```

Example 4–5 (Cont.): Getting Help for Mail Utility Commands

4.5.8 Exiting from the Mail Utility

Enter the **EXIT** command following the *MAIL>* prompt. When you enter the **EXIT** command, any messages in the *WASTEBASKET* folder are automatically deleted.

You can also exit from Mail by pressing CTRL/Z at the *MAIL>* prompt.

4.5.9 Using Folders to Organize Messages

All Mail files are subdivided into folders. By default, your Mail file (MAIL.MAI) contains a folder named *MAIL*. The *MAIL* folder contains messages you have already read. New messages automatically enter a folder named *NEWMAIL*. After the messages in the *NEWMAIL* folder have been read, they automatically move into the *MAIL* folder. The *NEWMAIL* folder automatically disappears after you have read all your new messages or exit from Mail.

You always know which folder you are currently in because the name of the folder is displayed at the top right corner of the screen when you enter the **READ** or **DIRECTORY** command.

You can create as many folders as you want by using the following commands:

- COPY
- FILE
- MOVE

Refer to the *VMS Mail Utility Manual* for descriptions of the above commands.

Figure 4–1 shows the relationship between a Mail message, a folder, and a file.

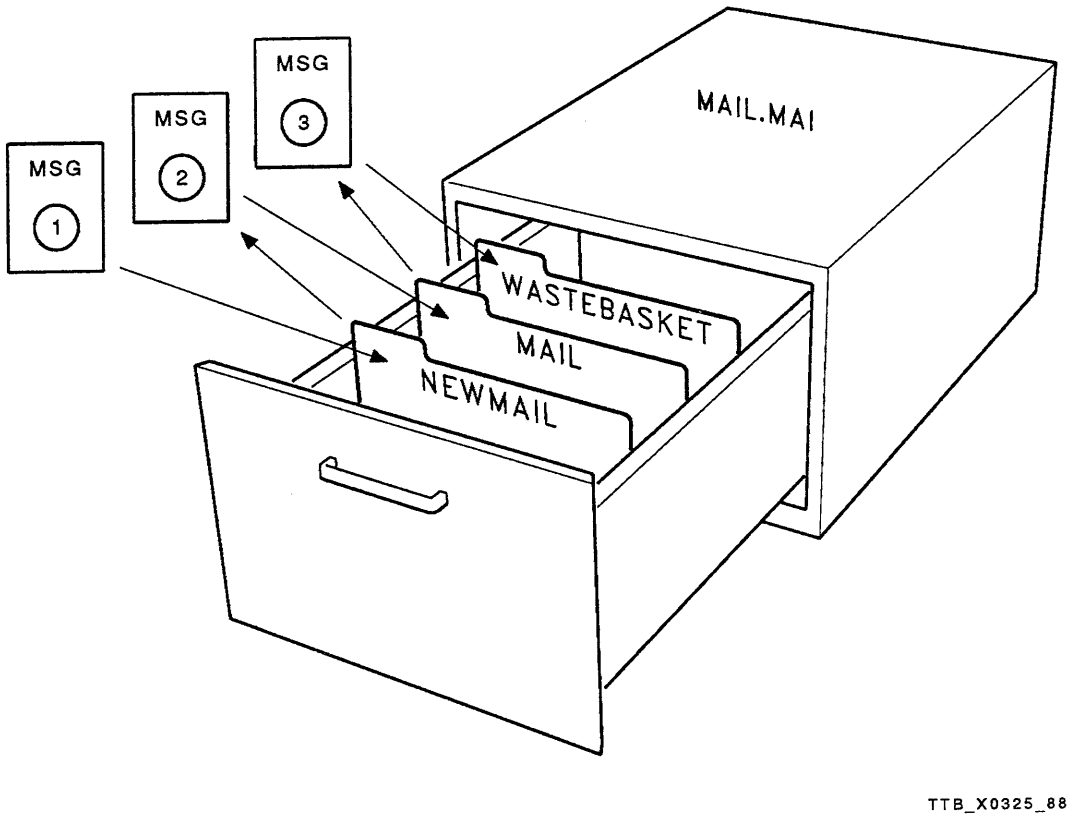


Figure 4–1 The Relationship Between a Mail Message, Folder, and File

Table 4–3 lists commands used to maintain messages.

Table 4–3 Mail Utility Commands Used to Maintain Messages

Operation	Format/Example	Comments
Displaying a list of folders	`DIRECTORY/FOLDER`	Displays a list of all folders in the current Mail file.
Displaying a list of messages	**DIRECTORY [folder–name]** `DIRECTORY CALENDAR`	Produces a list of all the messages in a folder. Each message contains an message number.
Moving between folders	**SELECT [folder–name]** `SELECT CALENDAR`	Moves you between folders of your choice.
Filing a message	**MOVE folder–name [file–name]** `MOVE CALENDAR`	Moves the current message to the folder named CALENDAR.
Copying a message to another folder	**COPY folder–name [file–name]** `COPY CALENDAR`	Places a copy of the current message into the specified folder.
Copying a message to a file	**EXTRACT file–name** `EXTRACT DWAYNE.TXT`	Places a copy of the current message into a sequential file with the file name specified.
Printing a message	`PRINT`	Places a copy of the current message into the default queue for printing. (Queues are presented in the Batch and Print module.)

4.6 THE PHONE UTILITY

The Phone utility allows you to communicate with users currently working on the system. The Phone utility, like the Mail utility, has a well–documented online Help facility and is invoked by entering the utility's name.

You can use the Phone utility to contact another user in either of the following ways:

- You can enter the name of a user as a parameter to the **PHONE** command.

```
$ PHONE HARKINS
```

- You can enter the name of a user from within the Phone utility.

```
$ PHONE
%DIAL HARKINS
```

The system responds to your request by displaying a repeating message on the terminal of user HARKINS. To respond to this repeating message, HARKINS must enter the Phone utility by entering the **PHONE** command. Then HARKINS can establish the terminal link by entering the **ANSWER** command, or reject it by entering the **REJECT** command.

After you enter the **PHONE** command, the system clears your terminal and displays a screen that has three sections. The top section echoes commands you enter. The middle section echoes the messages you send to other terminals in the conversation. The bottom section displays the messages you receive. Figure 4–2 illustrates the segmented terminal screen displayed by the Phone utility.

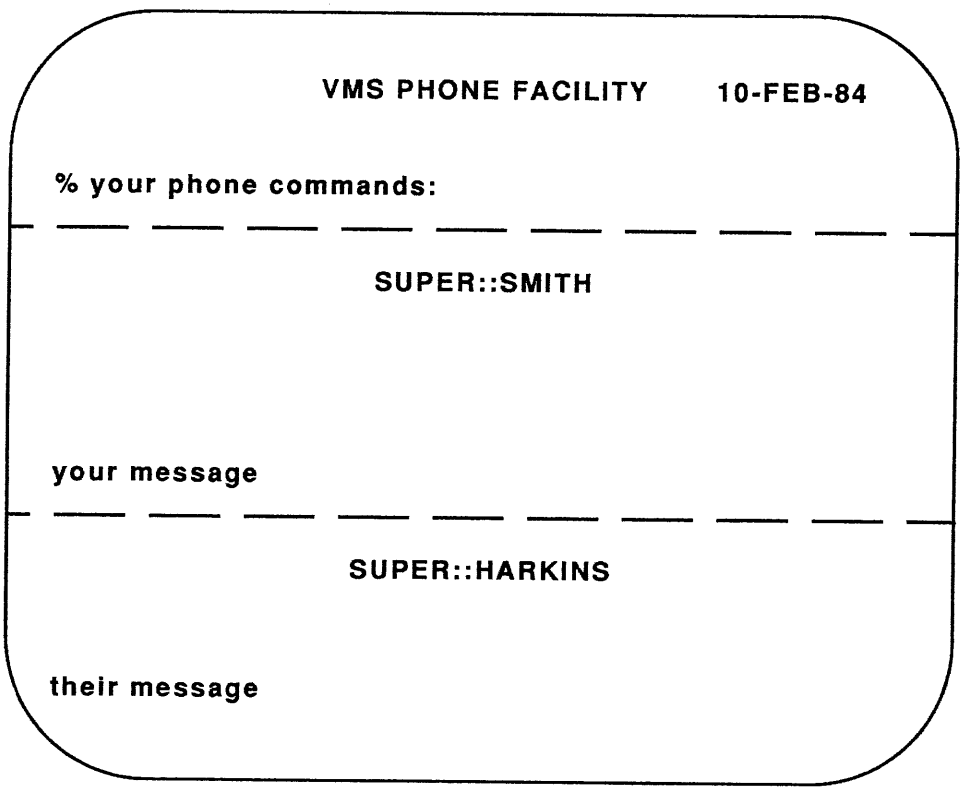


Figure 4–2 Using the Phone Utility

4.6.1 Getting Help on Phone Utility Commands

At any time during your Phone session, you can access the online Help facility by entering the **HELP** command. To display a particular help text, enter the command **HELP help–option**.

When your terminal is linked to the terminal of another, you must precede the **HELP** command, or any command to the Phone utility, with the percent sign (%). After you receive the Help display, type any character to refresh the Phone split screen.

Entering either the **EXIT** command or pressing CTRL/Z returns you to DCL command level. Remember, to enter the **EXIT** command during a conversation, you must first type a percent sign. Table 4–4 lists Phone commands used to create or reject a terminal link.

Table 4–4 Commonly Used Phone Utility Commands

Operation	Format/Example	Comments
Choosing who to call	**DIRECTORY [node[::]]** `DIRECTORY TRNTO::`	Displays a list of people with whom you could talk on your system or any other system in the network.
Requesting a terminal link	**DIAL user–name** `DIAL JONES`	Displays a repeating message on the terminal JONES is using that signifies a terminal link request.
Accepting a terminal link	`ANSWER`	Links your terminal to the terminal of the caller. The split screen is displayed.
Rejecting a terminal link	`REJECT`	Terminates the repeating message caused by the DIAL command entered by another user.
Placing others on hold	`HOLD`	Places all other terminals in the conversation on hold.
Reversing the hold	`UNHOLD`	Reverses your previously entered HOLD command.
Terminating a terminal link	`HANGUP`	Terminates the links of all terminals in the conversation.
Leaving the Phone utility	`EXIT` or `CTRL/Z`	Exits the Phone utility by first executing an automatic HANGUP command.

4.7 COMMUNICATING WITH OPERATORS

The DCL command **REQUEST** is usually used to communicate with system operators. The syntax of the **REQUEST** command is:

$ REQUEST "message-text"

The message text can have a maximum of 128 characters. If the text of the message contains more than one word, enclose the text in double quotation marks. The **REQUEST** command sends a message to all operators and allows you to continue to perform tasks at your terminal.

To send a message to an operator that requires a reply from that operator, use the **/REPLY** qualifier with the **REQUEST** command.

Note that if you request a reply from an operator, you will not be able to work on your terminal interactively until the reply is acknowledged by the operator.

You may wish then to cancel a **REQUEST/REPLY** message if no reply is sent by an operator. Press CTRL/C to interrupt the request and CTRL/Z to cancel it.

Example 4–6 illustrates a successful interchange between a user and an operator. Example 4–7 illustrates the procedure used to cancel a **REQUEST/REPLY** command.

```
$ REQUEST/REPLY "Please mount magtape 4 on MTA0"
%OPCOM-S-OPRNOTIF, operator notified waiting... 11:23:02.92
%OPCOM-S-OPREPLY, AFTER 11:30
13-MAR-1987 11:26:03.87, request 7 completed by operator OPA0
```

Example 4–6 Using the REQUEST/REPLY Command

```
$ REQUEST/REPLY "Please mount magtape"
%OPCOM-S-OPRNOTIF, operator has been notified, waiting... 10:28:02.48
CTRL/C
%OPCOM-I-RQST_PROMPT, REQUEST - Enter message or cancel request with control/Z.
REQUEST - Message?
CTRL/Z
%OPCOM-S-OPREPLY,
%%%%%%%%%%   OPCOM  13-FEB-1987 10:28:10.10  %%%%%%%%%%
Request 2305 was canceled
```

Example 4–7 Canceling a REQUEST/REPLY Command

4.8 SUMMARY

The Mail Utility

The Mail utility allows you to send to and receive messages from other users, both on your system and within a network. The Mail utility is invoked by entering **MAIL** at the DCL prompt.

- To read a new message, press RETURN at the *MAIL>* prompt.
- To read a message received while you are in the Mail utility, enter the **READ/NEW** command.
- To send a message, enter the **SEND** command at the *MAIL>* prompt.
  - — Enter the node (if different from your node) and user name.
  - — Enter the subject of the message.
  - — Enter the message.
  - — Press CTRL/Z after the last line of the message.

The Phone Utility

You can use the Phone utility to contact another user by:

- Entering the name of a user as a parameter to the **PHONE** command.
- Entering the name of a user from within the Phone utility.

To enter a command while having a conversation, type the percent sign (%) followed by the command.

The REQUEST Command

The **REQUEST** command displays a message at a system operator's terminal and optionally requests a reply.

```
REQUEST "message-text"

REQUEST/REPLY "message-text"
```

4.9 LABORATORY EXERCISE I

1. Invoke the Mail utility and send several Mail messages to someone.
2. Have your Mail recipient send messages to you.
3. Read your messages.
4. Obtain a list of your Mail messages.
5. Read only the second Mail message.
6. Delete the fourth Mail message.
7. Create a text file in your default directory. Send this file as a Mail message.
8. Pick a message and move it to a folder named *TEST*. Select the *TEST* folder and check to see if the message is there.
9. List the folders you have. In addition to the folder you just created, what other folders do you have?

4.10 LABORATORY EXERCISE II

1. Invoke the Phone utility.
2. Obtain a list of available users.
3. Establish a connection with one of the users.
4. Terminate all conversations.

4.11 LABORATORY EXERCISE III

Send a request to a system operator using the **REQUEST** command.

4.12 LABORATORY EXERCISE I—SOLUTIONS

1. Invoke the Mail utility and send several Mail messages to someone.

```
$ MAIL
MAIL> SEND
```

2. Ask your Mail recipient to send messages to you.

3. Read your messages.

```
MAIL> READ or
MAIL> 1 or
MAIL> (press RETURN)
```

4. Obtain a list of your Mail messages.

```
MAIL> DIRECTORY
```

5. Read only the second Mail message.

```
MAIL> READ 2
```

6. Delete the fourth Mail message.

```
MAIL> DELETE 4 or
MAIL> DELETE (if it is the current message on your screen)
```

7. Create a text file in your default directory. Send this file as a Mail message.

```
$ CREATE file-name
CTRL/Z
$ MAIL
MAIL> SEND file-name
```

8. Pick a message and move it to a folder named *TEST*. Select the *TEST* folder and check to see if the message is there.

```
MAIL> READ 1
MAIL> MOVE TEST
Folder TEST does not exist.
Do you want to create it (Y/N, default is N)? Y
%MAIL-I-NEWFOLDER, folder TEST created

MAIL> SELECT TEST
%MAIL-I-SELECTED, 1 message selected

MAIL> DIRECTORY
```

9. List the folders you have. In addition to the folder you just created, what other folders do you have?

```
MAIL> DIRECTORY/FOLDERS
```

You may possibly see three folders named *MAIL*, *NEWMAIL*, and *WASTEBASKET*.

4.13 LABORATORY EXERCISE II—SOLUTIONS

1. Invoke the Phone utility.

```
$ PHONE
```

2. Obtain a list of available users.

```
%DIRECTORY
```

3. Establish a connection with one of the users.

```
USER 1                  USER 2
%DIAL USER2             %ANSWER
```

4. Terminate all conversations.

%HANGUP *or* CTRL/Z

4.14 LABORATORY EXERCISE III—SOLUTION

Send a request to a system operator using the **REQUEST** command.

```
$ REQUEST "Please mount magtape"
```