Guide to VMS Files and Devices

Order Number: AA–LA06A–TE

April 1988

This guide describes procedures for using files and devices with the VMS operating system. The tasks described in this manual are oriented toward the use of private media.

Revision/Update Information:	This revised guide supersedes the *Guide to VAX/VMS Disk and Magnetic Tape Operations*, Version 4.4.
Software Version:	Version 5.0

digital equipment corporation
maynard, massachusetts

April 1988

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 postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

ZK3389

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire
03061

CANADA

Digital Equipment
of Canada Ltd.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

Digital Equipment Corporation
PSG Business Manager
c/o Digital§ local subsidiary
or approved distributor

In Continental USA and Puerto Rico call 800-258-1710.

In New Hampshire, Alaska, and Hawaii call 603-884-6660.

In Canada call 800-267-6215.

*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

Production Note

This book was produced with the VAX DOCUMENT electronic publishing system, a software tool developed and sold by DIGITAL. In this system, writers use an ASCII text editor to create source files containing text and English-like code; this code labels the structural elements of the document, such as chapters, paragraphs, and tables. The VAX DOCUMENT software, which runs on the VMS operating system, interprets the code to format the text, generate a table of contents and index, and paginate the entire document. Writers can print the document on the terminal or line printer, or they can use DIGITAL-supported devices, such as the LN03 laser printer and PostScript™ printers (PrintServer 40 or LN03R ScriptPrinter), to produce a typeset-quality copy containing integrated graphics.

™ PostScript is a trademark of Adobe Systems, Inc.

Contents

FIGURES

TABLES

Preface

The *Guide to VMS Files and Devices* describes some of the routine tasks that general users perform on files and devices.

Intended Audience

This guide is intended for all general users.

The *Guide to VMS Files and Devices* is not a guide for system managers; it is designed primarily for the private user. System managers should refer to *Introduction to VMS System Management* and *Guide to Setting Up a VMS System.*

If you are a novice user, you can use this guide as a stepping stone to a basic understanding of file operations on disk and magnetic tape devices. Experienced users unfamiliar with DIGITAL software can use it to gain familiarity with DIGITAL terms and techniques. Experienced DIGITAL users should find the guide useful because it describes in detail most of the disk and magnetic tape operations routinely performed on the VMS operating system.

Document Structure

The chapters in this guide contain the following information:

- Chapter 1 describes basic file and magnetic tape device concepts.
- Chapter 2 describes the protection schemes that apply to file and device media.
- Chapters 3-5 describe tasks frequently performed on private disk and tape media. These chapters include examples of command procedures designed to simplify the use of routine file and device operations.
- Appendix A describes the Files–11 On-Disk Structure (ODS–2) for VMS disk files and volumes.
- Appendix B describes formats for VMS magnetic tape volumes and files.

Associated Documents

The tasks described in this guide employ various VMS utilities and DIGITAL Command Language (DCL) commands. Detailed descriptions of the utilities and commands are provided in the VMS utilities volumes and in the *VMS DCL Dictionary.*

Your specific operations guide provides instructions for procedures used in system installation.

The *Guide to Setting Up a VMS System* provides task-oriented instructions for maintaining public files and volumes; these instructions include steps for bootstrapping and running standalone BACKUP.

The *VMS Local Area VAXcluster Manual* provides information for maintaining public files and volumes in a VAXcluster environment.

The *Guide to VMS System Security* describes security features available through the VMS operating system; see this guide for more information on data protection.

The *VAX Volume Shadowing Manual* (not part of the VMS document set) describes how to mount, dismount, and maintain volumes using the volume shadowing option.

Conventions

Convention	Meaning
RET	In examples, a key name (usually abbreviated) shown within a box indicates that you press a key on the keyboard; in text, a key name is not enclosed in a box. In this example, the key is the RETURN key. (Note that the RETURN key is not usually shown in syntax statements or in all examples; however, assume that you must press the RETURN key after entering a command or responding to a prompt.)
CTRL/C	A key combination, shown in uppercase with a slash separating two key names, indicates that you hold down the first key while you press the second key. For example, the key combination CTRL/C indicates that you hold down the key labeled CTRL while you press the key labeled C. In examples, a key combination is enclosed in a box.
$ SHOW TIME 05-JUN-1988 11:55:22	In examples, system output (what the system displays) is shown in black. User input (what you enter) is shown in red.
$ TYPE MYFILE.DAT . . .	In examples, a vertical series of periods, or ellipsis, means either that not all the data that the system would display in response to a command is shown or that not all the data a user would enter is shown.
input-file, . . .	In examples, a horizontal ellipsis indicates that additional parameters, values, or other information can be entered, that preceding items can be repeated one or more times, or that optional arguments in a statement have been omitted.
[logical-name]	Brackets indicate that the enclosed item is optional. (Brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
quotation marks apostrophes	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe (') is used to refer to a single quotation mark.

Summary of New and Changed Features

The *Guide to VMS Files and Devices* contains much of the same information as the Version 4.4 *Guide to VAX/VMS Disk and Magnetic Tape Operations.* The title and content have been modified to shift the focus to a more general user-oriented style.

Enhanced DCL Commands

The following DCL commands have been enhanced for VMS Version 5.0:

- ANALYZE/RMS_FILE/STATISTICS displays RMS file characteristics information for determining:
  - Whether journaling is enabled
  - The global buffer count
  - Whether file monitoring is enabled
- DIRECTORY/FULL displays whether journaling is enabled.
- SHOW DEVICES displays changes to system output.
- SHOW MAGTAPE displays change to odd parity line format.

1 Introduction to Files and Devices

This chapter provides an introduction to storage media and describes basic file and device concepts. It also discusses the importance of identifying routine operations performed on files and devices and the advantages of using command procedures to execute them.

1.1 The History of File Storage Media

As technology has progressed, so has the volume of information that must be saved. Business and industrial concerns, for instance, have compiled many types of data files about a wide range of subjects. For many years, all these data files were stored on paper in desk drawers and filing cabinets. But, as these paper files grew, it often took longer to locate the needed data than to create it in the first place. Storing all of a company's data files on paper became impractical.

As the need for saving data increased, the need also arose for a better medium for storing and retrieving data quickly, reliably, and economically. The computer and computerized filing systems provide such storage media.

At first, computerized files consisted of collections of punched cards, which provide a means of grouping related pieces of information. This information might represent, for example, a business event such as a purchase or sale of office furniture. Or, in the engineering environment, the information could represent variable equations and data constants related to stress analysis. This information, grouped on a single card, represents a record of that event. Records of similar events, grouped together, constitute a file.

As a storage medium, cards have certain advantages. They are easy to add, delete, or rearrange. However, cards become worn, require physical handling, and are bulky. Cards are also relatively slow to process because they allow only sequential access. Sequential access means that the search for a record starts at the beginning of the file and proceeds in order through each record. At times, when the needed record (or group of records) is near the end of the file, the search wastes computer processing time.

The introduction of magnetic tape as a storage medium eliminated some of the disadvantages of cards. Magnetic tape provides both a storage medium and a means for input/output (I/O), and its uses have grown along with the needs of the user. Magnetic tape offers virtually unlimited storage. It requires much less storage space than a card file with a comparable amount of data. However, magnetic tape is limited as a storage medium because, like a card file, it allows only sequential access.

In contrast to magnetic tape, disk storage allows direct file access. Direct access means that the computer can locate the desired record without first searching the records that precede it in the file. The time needed to access the record is independent of the record's location in the file.

1.2 Basic Device Concepts

This section describes basic disk and magnetic tape device concepts for VMS systems.

1.2.1 Disk Concepts

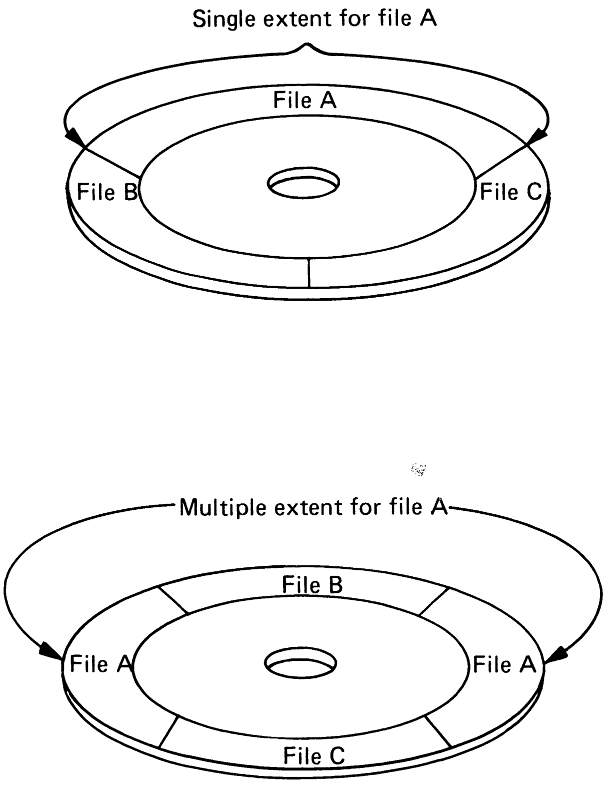
VMS files reside on Files–11 On–Disk Structure volumes. The term *Files–11 On–Disk Structure* refers to the logical structure given to the disk; namely, a hierarchical organization of files, their data, and the directories needed to gain access to them. The VMS file system implements the disk structure and provides access control to the files located on the disk. This section describes the Files–11 On–Disk Structure levels and defines the terminology related to it. (The term *Files–11* used alone always refers to *File–11 On–Disk Structure levels.*)

The smallest addressable unit of information on a disk is a block. Files–11 On-Disk Structures define a block to consist of 512 8-bit bytes. Blocks can be treated as units for transfer between a Files–11 disk volume and memory.

Blocks are logically grouped into clusters, which are the basic unit by which disk space is allocated. The system manager or operator determines the number of blocks in a cluster when a given disk, known as a volume, is first prepared for use (initialized). Cluster sizes from 1 to 65,535 blocks are allowed, but the smaller cluster sizes in the range are more practical. In general, a disk with a relatively small number of blocks is given a smaller cluster size, while larger disks are given larger cluster sizes to minimize the overhead for disk space allocation.

Contiguous clusters allocated to a particular file have been given the name *extent.* An extent can contain all or part of a file. If enough contiguous area is available on the disk, the entire file is allocated as a single extent. Sometimes, however, not enough contiguous area is available to contain the entire file, or, when you create a file initially, you may not wish to reserve the entire required amount of space. When the file is eventually extended, it is unlikely that the adjacent clusters will still be unallocated. If the adjacent clusters are already allocated to another file, the extension will not occur contiguously. Whether the clusters are contiguous or not, the file is divided into two or more parts, and each part is an extent. Thus, a file can consist of multiple extents located in separate areas on the disk, as shown in Figure 1–1. Note that the file extensions are done automatically.

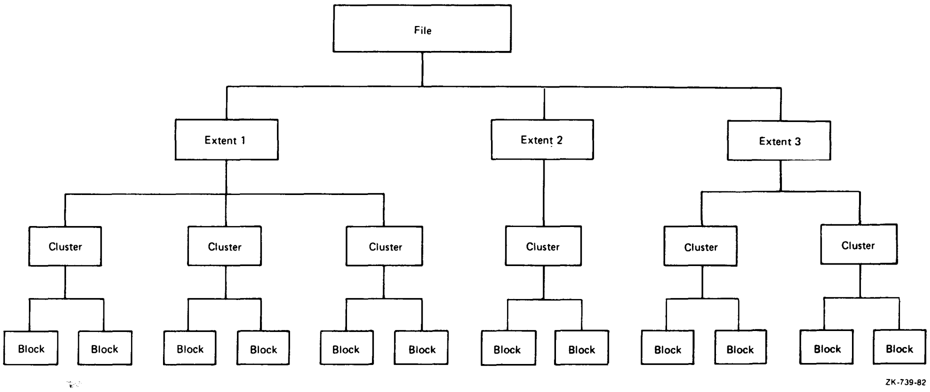
Figure 1–1 File Extents



ZK-738-82

Figure 1–2 shows the hierarchy of blocks, clusters, extents, and files in the Files–11 On-Disk Structure.

Figure 1–2 Files–11 On-Disk Structure Hierarchy



You can exercise as little or as much control as you want over the allocation of space on a Files–11 disk file. For example, you can specify the number of blocks to be allocated and even give the exact location for the blocks on the volume. On the other hand, you can allow VMS Record Management Services (RMS) to handle all disk space allocation details automatically.

If you want, you can specify the size of the initial space allocation and the size to be used by VMS RMS each time the file is extended. If you find that a file needs less space than is allocated to it, you can specify that the unused clusters are to be deallocated from the file. These clusters will then be available for allocation to other files on the volume.

When a large amount of file storage space is needed, you can combine several Files–11 volumes into what is called a volume set. A volume set, although composed of several physical volumes, has the appearance of one large volume. The different extents of a file can be located on different volumes in the volume set. In general, you need not specify a particular volume in the set to locate a file or create a new one, although it is possible (and sometimes desirable for performance reasons) to specify a particular volume to be used for a certain allocation request.

The smallest unit discernible to the Files–11 structure is the sector; for most Files–11 disks, a sector is equivalent to a block, which is 512 bytes. Other basic terms related to disks are track and cylinder. A track is the collection of sectors (or blocks on Files–11 structures) at a single radius on one recording surface of a disk. It is accessible to a given read/write head position on the disk device. A cylinder consists of all tracks at the same radius on all recording surfaces of a disk.

Because access to any of the blocks in a given cylinder does not require any movement of the disk's read/write heads, it is generally advantageous to keep related data blocks in the same cylinder. For this reason, when choosing a cluster size for a large-capacity disk, a system manager often selects a cluster size that divides evenly into the cylinder size.

Figure 1–3 Tracks and Cylinders

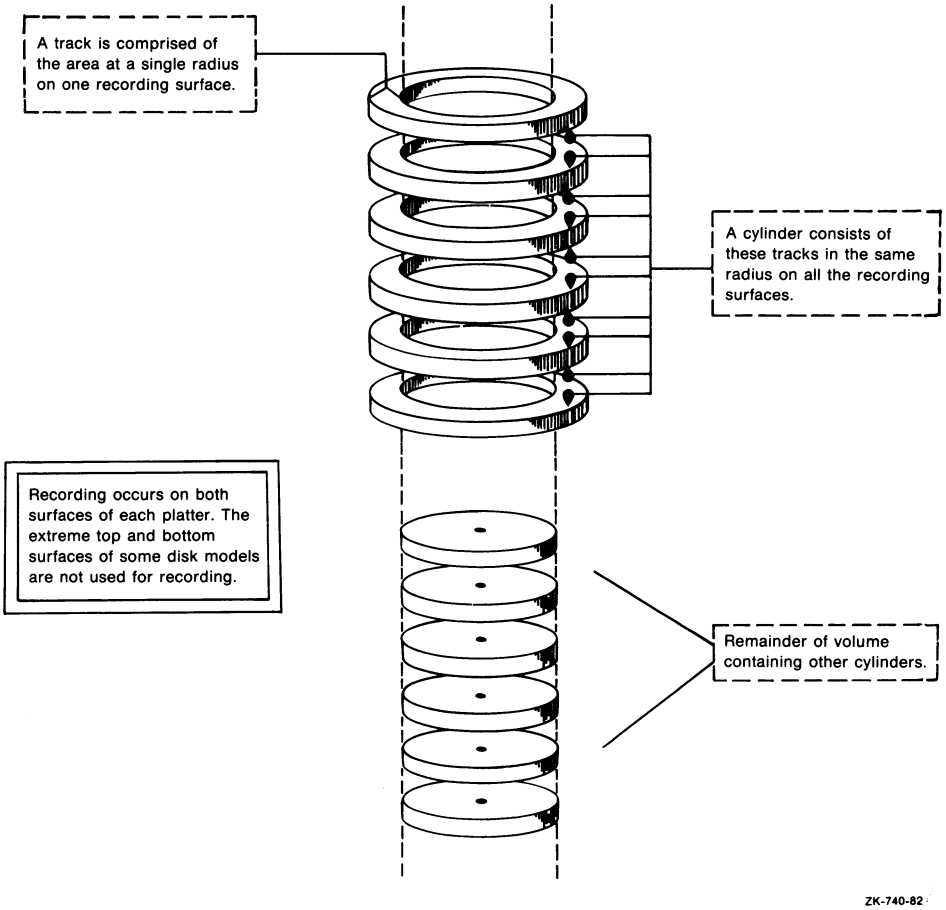

A track is comprised of the area at a single radius on one recording surface.
A cylinder consists of these tracks in the same radius on all the recording surfaces.
Recording occurs on both surfaces of each platter. The extreme top and bottom surfaces of some disk models are not used for recording.
Remainder of volume containing other cylinders.
ZK-740-82

Figure 1–3 is a graphic representation of tracks and cylinders.

The remainder of this section contains a brief explanation of some basic elements of the Files–11 structure.

A Files–11 structure resides on a volume, which is a physical medium such as a disk pack. A Files–11 volume is an ordered set of 512-byte blocks. The blocks are numbered consecutively from 0 to *n—1;* the value of *n—1* is the size of the disk in blocks.

Each Files–11 volume has an index file, which is created when the volume is initialized. (You cannot use a disk as a Files–11 disk until it has been initialized with the INITIALIZE command.) The index file contains the following information:

- Bootstrap block
- Home block
- File headers

The *bootstrap block* contains the bootstrap program and is physically the first block on the volume. All Files–11 volumes have an area for this bootstrap block even if the operating system does not require a bootstrap block. If the volume is not the operating system volume, the bootstrap block area contains a program other than the bootstrap program. If an attempt is made to boot a nonsystem disk, this program displays a message on the system console indicating that the volume is not the operating system volume.

The *home block* is normally the next block after the bootstrap block; it identifies the disk as a Files–11 volume. If for some reason the home block cannot be read (physically unusable), an alternative block will be selected for use as the home block. This block provides specific information about the volume and default values for files on the volume. Among the items in the home block are the following:

- The volume name
- Information to locate the remainder of the index file
- The maximum number of files that can be present on the volume at any one time
- The user identification code (UIC) of the owner of the volume
- Volume protection information (specifies which users can read or write the entire volume)

Files–11 volumes contain several copies of the home block to ensure against accidental destruction of this information and the consequent loss of access to files on the volume.

The bulk of the index file consists of *file headers*; each file header describes one file on the volume. File headers contain information such as the owner UIC, protection, creation date and time, and Access Control Lists (ACLs). Most important, the file header contains a list of extents that make up the file, describing where the file is logically located on the volume. If a file has a large number of extents, multiple file headers may be used to describe them. A file identifier number is associated with each file header.

When you create a file, you normally specify a file name to VMS RMS, which assigns this name to the file on a Files–11 volume. RMS places the file name and file identifier associated with the newly created file in a directory, which contains an entry defining the location for each file. When you access the file, you supply the file name, which supplies a path to the file identifier through the directory entry. The file identifier, in turn, points to the location of the file header, which contains a listing of the extent or extents that locate the actual data.

1.2.2 Magnetic Tape Concepts

The VMS file storage system for magnetic tapes is based on the standard magnetic tape structure as defined by the American National Standard X3.27-1978 (referred to as the ANSI standard throughout this manual) and also supports the ISO 1001-1979 standard.

Magnetic tape data is organized sequentially; the data records and files are organized in the order in which they are written.

On VMS systems, characters of data on magnetic tape are most commonly measured in bits per inch (bpi). This measurement is called density. (The ANSI standard uses characters per inch (CPI), which is equivalent to the bpi convention used by VMS.) A 1600-bpi tape can accommodate 1600 characters of data, using an inch of tape.

Even though a magnetic tape may have a density of 1600 bpi, there are not always 1600 characters on every inch of tape because of the interrecord gap (IRG). The IRG is an interval of blank space between data records; it is created automatically when records are written to the magnetic tape. The IRG is a breakpoint on the magnetic tape, which allows the magnetic tape unit to decelerate and stop, if necessary, after a record operation and then accelerate to working speed for the next record operation.

Each IRG is approximately 0.6 inch in length (this length varies with the type of tape drive). Writing an 80-character record at 1600 bpi requires 0.05 inch of space. Therefore, the IRG requires 12 times more space than the data, wasting valuable storage space. RMS can reduce the size of this wasted space by using record blocking. This technique groups individual records into a block and places the IRG after the block rather than after each record.

Note that a block on magnetic tape is different from a block on disk. On disk, a block is fixed at a size of 512 bytes; on magnetic tape, the size of a block is determined by the user. However, record blocking requires more buffer space to be allocated for your program, which increases RMS memory requirements. The greater the number of records in a block, the greater the buffer size requirements. You must determine the point at which the benefits of record blocking cease. This determination should be based on the configuration of your computer system and your authorized environment.

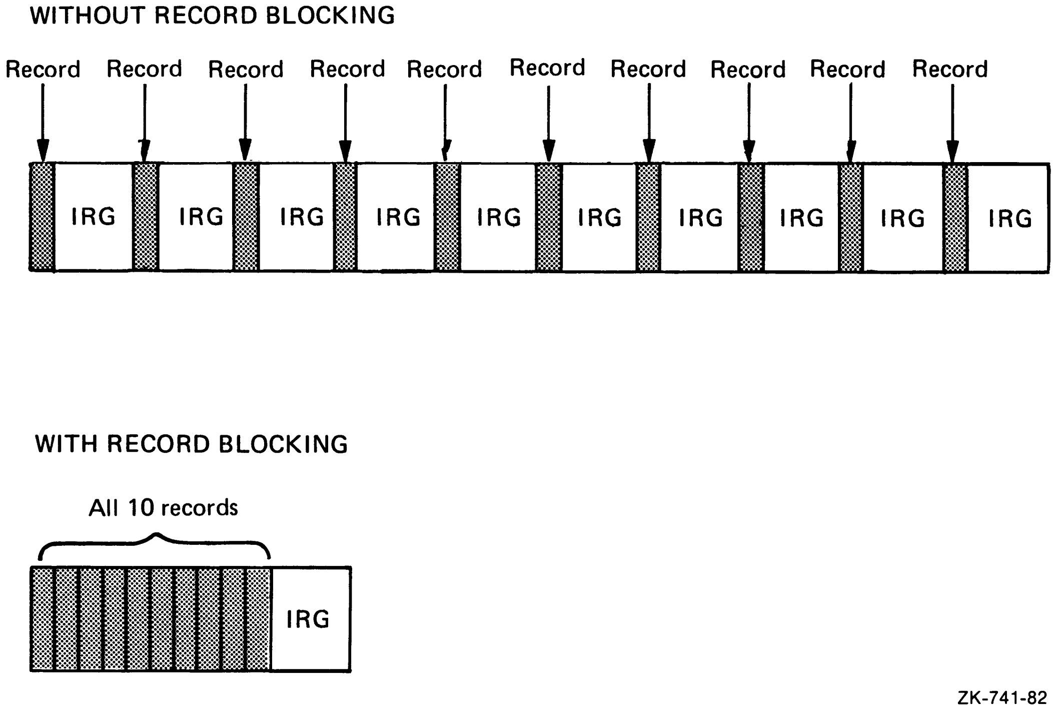
Figure 1–4 shows how space can be saved by record blocking. Assume that a 1600-bpi magnetic tape contains 10 records that are not grouped into a block. Each record is 160 characters long (0.1 inch at 1600 bpi) with a 0.6 inch IRG after each record. This uses 7 inches of tape. However, placing the same 10 records into one block uses only 1.6 inches of tape (1 inch for the data records and 0.6-inch for the IRG).

Record blocking also increases the efficiency of the flow of data into the computer. For example, ten unblocked records require ten separate physical transfers, while ten records placed into a single block require only one physical transfer. Moreover, a shorter length of magnetic tape is traversed for the same amount of data; thus, the operation is completed in less time.

Data on magnetic tape is also organized into files. When you create a file on magnetic tape, the magnetic tape file system writes a set of header labels on the tape immediately preceding the data blocks. These labels contain information such as the user-supplied file name, creation date, and expiration date. Additional labels, called trailer labels, are also written following the file. To access a file on magnetic tape by the file name, the file system searches the tape for the header label set that contains the specified file name.

When the data blocks of a file or related files do not physically fit on one volume (a reel of magnetic tape), the file is continued on another volume. This is a multivolume file. The volumes in a multivolume file constitute a volume set. When access is made to a file on a volume set, all volumes in the set are accessible.

Figure 1–4 Interrecord Gaps



1.3 Using Command Procedures to Perform Routine File and Device Operations

Many of the operations that you perform on disk and magnetic tape media are routine in nature. Therefore, you will find it worthwhile to take the time to identify those tasks that you routinely perform at your particular site. Once you have isolated those tasks, you can design command procedures to assist you in performing them.

If you are a system manager or an operator, for example, you must frequently perform data integrity tasks such as backing up media. You could enter all of the commands, parameters, and qualifiers required to back up your media each time that you perform the backup operation, or you can write a single command procedure (containing that set of commands, qualifiers, and parameters) that, when executed, would also perform the backup operation.

In order to familiarize yourself with the syntax used to design and execute command procedures, see the *Guide to Using VMS Command Procedures.*

2 File and Device Protection

The VMS operating system provides various types of protection. This chapter focuses on data, file, and device protection for VMS systems.

2.1 Data Protection

The VMS operating system protects data on disk and magnetic tape volumes to ensure against accidental or unauthorized access. The term *volume* refers to the entity that exists when a medium is mounted on a device. For example, disk packs and reels of tape are called volumes when they are mounted on disk and magnetic tape drives.

VMS systems support the protection of data on disk and magnetic tape media at the volume and file levels. At the volume level, the system provides protection for both disks and tapes. At the file level, the system provides protection for individual disk files, directory files that reside on disk volumes, and tape files. However, distinct file-level protection for magnetic tape files is limited to one special case, described in Section 2.2.2.4.

In addition to protecting the data on mounted volumes, the system also provides device-level protection. For more information on setting device protection characteristics, see the descriptions of the DCL commands INITIALIZE, MOUNT, and SET VOLUME in the *VMS DCL Dictionary*.

On VMS, data residing on disk and tape volumes can be protected by one or more of the following:

- User identification codes (UICs)
- Access control lists (ACLs)
- ANSI-standard accessibility (magnetic tape only)

2.1.1 User Identification Code (UIC)-Based Protection

User Identification Code (UIC)-based protection is supported for disk and magnetic tape volumes and for individual files on disk volumes, including directories. UIC-based protection is determined by an owner's user identification code and a protection code. The owner UIC is normally the UIC of the user who created the file or volume. The protection code indicates who is allowed access and for what purposes.

UIC formats can contain alphanumeric characters or they can consist entirely of octal numbers. For example, [VMS,USER] and [360,030] are both legitimate UIC formats.

When a user attempts to access a file or volume, the user's UIC is compared against the owner UIC of the file or volume. Depending on the relationship of the UICs, the user falls into one or more of the following categories:

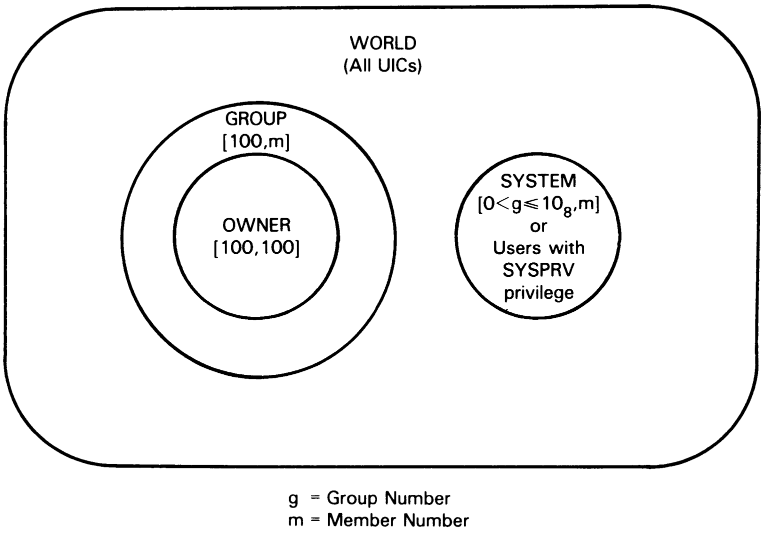
- SYSTEM — All users who have system privilege (SYSPRV) or low group numbers, usually from 1 through 10 (octal). However, the exact range of system group numbers is determined by the system manager when the system is generated and may range as high as 37776 (octal). These group numbers are generally for system managers, system programmers, and operators. In addition, the SYSTEM protection field is used for those users with the user privilege GRPPRV, whose UIC group matches the group of the file or volume owner UIC.

- OWNER — The user with the same UIC as the person who created, and therefore owns, the volume or file
- GROUP — All users, including the owner, who have the same group number in their UICs as the owner of the file
- WORLD — All users including those in the first three categories

Note: **SYSTEM and OWNER categories always have access to magnetic tape volumes, except as noted in Section 2.1.3.**

Figure 2–1 illustrates the relationships of these categories to each other.

Figure 2–1 Illustrating User Categories with a UIC of [100,100]



NOTE: THE SYSTEM MANAGER CAN EXTEND THE SYSTEM GROUP NUMBER LIMIT TO 37776$_8$

ZK-778-82

Each user category is allowed or denied five types of access:

- READ
- WRITE
- EXECUTE
- DELETE
- CONTROL

The meaning of each access type depends on the object to which it is applied (disk or magnetic tape volume, file, or directory). Section 2.2.1 describes how these types of access apply to disk and magnetic tape volumes; Section 2.2.2 describes how the access types apply to files and directories.

Note that CONTROL access is not explicitly set with the SET PROTECTION command. CONTROL access to a file is a function of ownership and privilege; therefore, it is never granted to the GROUP and WORLD categories of users, but is always granted to SYSTEM and OWNER.

You can bypass all UIC-based protection checks if you have the BYPASS user privilege. To bypass all protection checks and gain READ and CONTROL access to a disk file, you need READALL user privilege. For magnetic tapes, you can bypass UIC-based protection checks if you have VOLPRO (volume protection override) privilege.

For information on how to set UIC-based protection, see the discussion of protection codes in the *VMS DCL Concepts Manual.*

2.1.2 Access Control List (ACL)-Based Protection

An access control list (ACL) is primarily a list of entries that grant or deny access to a particular system resource, such as a disk file or a directory. Each access control list consists of one or more entries known as access control list entries (ACEs).

ACLs offer you the chance to "fine tune" the action taken when access to a file is sought. You can opt to provide an ACL on any disk file to permit as much or as little access as is desirable in each case. In providing a more detailed definition of who is allowed a particular kind of access, ACLs can enhance the security of disk files. Note, however, that "fine tuning" is done at the expense of performance; the larger the ACLs are, the more time they take to process.

ACL-based protection is supported for files and directories on disk volumes only. ACL protection is not supported for files on magnetic tape volumes.

If a file does not have an associated ACL, the system uses UIC-based protection to determine access as described in Section 2.1.1.

If a file has an associated ACL, the system uses the ACL to determine access as follows:

- If the ACL includes an identifier entry for a particular UIC, the ACL specification takes precedence over your UIC-based protection. However, SYSTEM and OWNER may still have access. For detailed information on ACL protection, see the *Guide to VMS System Security.*
- If the ACL does not include an identifier entry for your UIC (that is, does not explicitly allow or refuse access), then the system uses UIC-based protection to determine access.

If you have CONTROL access to a resource, you can define the access control list using the DCL commands EDIT/ACL, SET ACL and SET FILE/ACL.

For more information on how to invoke, modify, and display ACLs, see the *VMS Access Control List Editor Manual* or the *Guide to VMS System Security.*

2.1.3 VMS ANSI-Labeled Magnetic Tape Accessibility Protection

The VMS magnetic tape file system supports accessibility protection based on the ANSI and ISO standards. This protection scheme allows an installation to use a routine that is designed to interpret the contents of the volume- and header-label accessibility field. See the $MTACCESS system service in the *VMS System Services Reference Manual* for more information on installation routines.

When the installation routine is called to interpret a VOL1 label supplied as input, the routine must return one of the following values to the magnetic tape file system:

- User has no access to the volume or file
- User has complete access to the volume or file
- The output of any other VMS protection mechanism specified (for file-level access, the protection defaults to whatever was specified for the volume)

If the installation routine determines that you have no access to a volume or file, then you will be denied access regardless of the volume protection set by VMS. That is, no privilege granted to you by VMS can override that decision. On the other hand, if the installation determines that you have complete access to the file or volume, then you will be granted complete access even if a VMS protection scheme denies you access to that file or volume.

When the installation routine is called to return the VOL1 label or HDR1 label as output, the routine must return the character to write into the volume- or file-label accessibility field.

If you do not design your own installation routine, the VMS operating system provides a default routine for you, which works in the following way. The default installation routine first checks the ANSI standard version of the label. For magnetic tapes with a version number of 3 or less, the routine outputs either an ASCII character space (full access without checking VMS protection) or the character specified by the user. On input of these magnetic tapes, the routine checks for a blank (full access without checking VMS protection) and, if the field is not blank, returns the value that causes the file system to check for explicit override of accessibility checking.

For magnetic tapes with a version number greater than 3, the routine outputs either the character specified by the user or an ASCII 1 if no character was specified. On input of these magnetic tapes, the routine checks for an ASCII character space. If the field has an ASCII space, the user is given full access and VMS protection is not checked. If the field contains an ASCII 1 and the VMS protection has been specified, the VMS protection is checked.

If the field does not contain an ASCII space or an ASCII 1, the routine returns the value to the magnetic tape file system that forces the user to override the accessibility checking and allows the magnetic tape file system to check VMS protection.

When called to return a VOL1 or HDR1 label as output, the routine returns the ASCII character 1 if a VMS protection characteristic has been specified for the volume. If no VMS protection was specified, then the routine returns the ASCII character space.

When the installation routine is called to interpret a VOL1 or HDR1 label supplied as input, the routine returns the value that causes the magnetic tape file system to check the VMS protection under the following conditions:

- If the routine finds an ASCII 1 in the accessibility field when a VMS protection has been specified and the magnetic tape conforms to an ANSI standard greater than Version 3
- If the magnetic tape conforms to an ANSI standard greater than Version 3
- An ASCII space for all other cases

For all other conditions, the routine provided by VMS returns the value to the magnetic tape file system that causes the file system to check for explicit override of accessibility processing.

For more information on how ANSI-standard accessibility protection applies to files on magnetic tape volumes, see Section 2.2.2.4.

2.2 File Protection

This section discusses file protection at both the volume level and file level.

2.2.1 Volume-Level Protection

Disks and tapes can both be protected at the volume level. When you prepare a disk or magnetic tape volume for private use, you can define the protection you want applied.

Volume protection for a disk or magnetic tape volume is usually set when the volume is mounted or initialized. If you do not explicitly specify the protection for a particular disk volume, the system provides a default protection for you.

To change the protection that has been set on a disk volume, use the DCL command SET VOLUME as described in the *VMS DCL Dictionary.* For more information on setting the volume protection when you mount a volume, see the *VMS Mount Utility Manual* and the descriptions of the DCL command INITIALIZE in the *VMS DCL Dictionary.*

2.2.1.1 Disk Volume Protection

Disk volume protection supports the following access types:

- READ
- WRITE
- EXECUTE
- DELETE

If you have READ access to a disk volume, you have the right to examine, print, execute, or copy files from that volume.

If you have WRITE access to a disk volume, you have the right to modify existing files on that volume. Unlike that of magnetic tape volumes, WRITE access to a disk volume does not automatically imply READ access. That is, it is possible to grant a user WRITE access to a disk volume without granting that user READ access to the volume.

When applied at the volume (as opposed to the file) level, EXECUTE access means that you have the right to create files or modify existing files. When applied at the file level, EXECUTE access gives you the right to invoke executable files (such as executable images and command procedures).

If you have DELETE access to a disk volume, you have the right to delete files on the volume.

2.2.1.2 Protection of Disk Volumes

By default, no protection is applied to newly initialized disk volumes. You can specify protection with the /PROTECTION qualifier of the INITIALIZE command, and you can specify an ACL for a disk volume. The following example specifies UIC-based protection for the disk volume ACCOUNT1:

```
$ INITIALIZE WORKDISK: ACCOUNT1 -
_$ /PROTECTION=(S:RWED,O:RWED,G:R,W:R)
```

You can respecify the protection each time you mount the volume by using the /PROTECTION qualifier of the MOUNT command. You must own the volume or have VOLPRO privilege to change protection.

You can also limit access to a disk volume with the following qualifiers to the INITIALIZE and MOUNT commands:

- /SYSTEM — All processes have RWED access to the volume, but only system processes (or processes with SYSNAM and SYSPRV privileges) can create first-level directories. (The volume is owned by [1,1].)
- /GROUP — System, owner, and group processes have RWED access to the volume. World users have no access.
- /NOSHARE — System and owner processes have RWED access to the volume. Group and world users have no access.

At initialization time, the above qualifiers override any protection mask specified. At mount time, however, the protection mask overrides the qualifiers. When mounting or dismounting a volume, you must have GRPNAM privilege to specify /GROUP and SYSNAM privilege to specify /SYSTEM.

2.2.1.3 Magnetic Tape Volume Protection

Unlike disk volume protection, only READ and WRITE access apply to magnetic tape volume protection.

If you have READ access to a magnetic tape volume, you have the right to examine, print, or copy files from the volume. If you have WRITE access to a magnetic tape volume, you have the right to append or write files to the volume.

For magnetic tape volumes, WRITE access implies READ access. Thus, granting a category of users WRITE access to a magnetic tape volume automatically permits them to have READ access to the volume also.

There are two ways to protect a magnetic tape volume on a VMS system. You can protect a magnetic tape volume through the UIC-based protection scheme supported by VMS software. This scheme is checked on VMS systems only and will be ignored in any interchange with non-VMS systems. You can also protect a magnetic tape volume by using the guidelines of the ANSI standards. This protection scheme supports the protection of magnetic tape volumes in environments where interchange exists between VMS and non-VMS operating systems.

2.2.1.3.1 Protection for VMS Magnetic Tapes

The VMS magnetic tape file system provides two levels of protection. One level is defined by the ANSI standard and is encoded in the ACCESSIBILITY field of the first volume label written on the magnetic tape. The second level is UIC-based and is defined by the VMS magnetic tape file system. This protection is encoded in the second volume label written on the magnetic tape.

2.2.1.3.2 Protection for Interchange Environments

Magnetic tape volume protection is also supported for interchange between VMS and non-VMS operating systems. Protection is supported for interchange between VMS and DIGITAL operating systems other than VMS (such as RSX–11M and TOPS–20) as well as between VMS and non-DIGITAL operating systems.

For magnetic tapes processed on any operating system that supports a version of the ANSI standard later than Version 3, the accessibility information in the first volume label is processed exactly as described above. Magnetic tapes processed on operating systems other than VMS Version 4.0 or later have their protection characteristics encoded in the first volume label of the magnetic tape volume.

In order to process a magnetic tape created on a DIGITAL operating system other than VMS, a user must have VOLPRO privilege and must explicitly override the check on the protection. If the magnetic tape had been created with a specified accessibility, then a user wishing to access that tape must have the appropriate privilege and must explicitly override the check on accessibility. If the magnetic tape volume had not been created with such a protection scheme, then a user wishing to access that magnetic tape would be granted READ and WRITE access to that magnetic tape volume.

The VMS magnetic tape file system allows you to specify values for the fields in which other DIGITAL operating systems currently write their protection information. Except under conditions described in the two previous sections, VMS will not process this field. Thus, this field can be used to store the protection values for another operating system without affecting the VMS protection characteristics on that particular volume.

2.2.2 File-Level Protection

In addition to volume-level protection, the VMS operating system supports protection at the file level for all files residing on disk volumes, including directories. In certain cases, file-level protection is also supported for magnetic tape files. For files residing on magnetic tape volumes, however, only one special case (described in Section 2.2.2.4) exists in which distinct file-level protection applies.

File-level protection is described as it applies to each of the following:

- Disk file protection
- Directory file protection
- Magnetic tape file protection

2.2.2.1 File Protection

For the most part, file protection is transparent. The tools exist, however, to adjust the protection of a file as you see fit. You must own the file; have control access to the file; or have GRPPRV, SYSPRV, BYPASS, or READALL privilege to set the protection or modify the ACL of a file.

Note: **You cannot completely protect a file without applying at least the same protection to the directory in which the file resides; see Section 2.2.2.3.1 for information on directory protection.**

2.2.2.1.1 Default File Protection

A new file receives default UIC-based protection and the default access control entries (ACEs), if any, of its parent directory. A renamed file's protection is unchanged. A new version of an existing file receives the UIC-based protection and ACL of the previous version. (Use the /PROTECTION qualifier of the BACKUP, COPY, and CREATE commands to override the default UIC-based protection.)

- Default UIC protection — The operating system provides each process with a default UIC-based protection of (S:RWED,O:RWED,G:RE,W). To change the default protection, enter the SET PROTECTION command with the /DEFAULT qualifier. For example, if you place the following command in your login command procedure, you grant all processes read and execute access to any files that you subsequently create:

  ```
  $ SET PROTECTION = (S:RWED,O:RWED,G:RE,W:RE)/DEFAULT
  ```

  (Remember that you must execute the login command procedure for this command to execute.)

- Default ACL protection — You can override default UIC protection for specified directories or subdirectories by placing a default-protection ACE in the ACL of the appropriate directory file. The default protection specified in the ACE is applied to any new file created in the specified directory or in any subdirectory of the directory. The following ACE, which must be in the ACL of a directory file, specifies that the default protection — for that directory and for the directory's subdirectories — allows system and owner processes full access, group processes read and execute access, and world users no access:

  ```
  (DEFAULT_PROTECTION,S:RWED,O:RWED,G:RE,W:)
  ```

To specify a default identifier ACE to be copied to the ACL of any file subsequently created in the directory, specify the DEFAULT option in the directory file's identifier ACL.

2.2.2.1.2 Explicit File Protection

You can explicitly specify UIC-based protection for a new file with the /PROTECTION qualifier (valid with the BACKUP, COPY, RENAME, and CREATE command), as demonstrated in the following command line:

```
$ CREATE MAST12.TXT/PROTECTION=(S:RWED,O:RWED,G,W)
```

You can change the UIC-based protection on an existing file with the SET PROTECTION command, as follows:

```
$ SET PROTECTION=(S:RWED,O:RWED,G,W) MAST12.TXT
```

After a file is created and you have created an ACL for the file, you can modify the ACL and add as many ACEs to the ACL as you desire. The protection specified by the ACL overrides the file's UIC protection.

2.2.2.2 Disk File Protection

Each file on a disk has its own protection code, which is distinct from the protection that applies to the disk volume itself. For files residing on disk volumes, the following access types are supported:

- READ
- WRITE
- EXECUTE
- DELETE
- CONTROL

If you have READ access to a file or group of files on a disk volume, you have the right to examine, print, or copy that file or group of files. READ access automatically includes EXECUTE access to a specified file or group of files on disk.

If you have WRITE access to a file or group of files on a disk volume, you have the right to modify the file or group of files. As with disk volume protection, it is possible to be granted WRITE access without having READ access. This is not very useful, however, since opening a file for a write operation implies READ access.

If you have EXECUTE access to a disk file or group of disk files, you have the right to execute executable program images or DCL command procedures contained in that file or group of files.

If you have DELETE access to a particular disk file or group of disk files, you have the right to delete the file or group of files.

If you have CONTROL access to a particular disk file or group of disk files, you have the right to change the characteristics of the file or group of files.

You can specify a protection code when you create or copy a file by using the /PROTECTION qualifier, as in the following example:

```
$ COPY/PROTECTION=(SYSTEM:RW,OWNER:RWED,GROUP:RW,WORLD) ABC.DAT
```

You can also change the protection for an existing file by using the SET PROTECTION command. For example, enter the following:

```
$ SET PROTECTION=(SYSTEM:RWE,OWNER:RWED,GROUP:RE,WORLD) ABC.DAT
```

This command gives the VMS operating system the following instructions regarding the file ABC.DAT: SYSTEM has READ, WRITE, and EXECUTE privileges; OWNER has READ, WRITE, EXECUTE, and DELETE privileges; GROUP has READ and EXECUTE privileges only; and WORLD has no access.

For SYSTEM and OWNER, CONTROL access is implied and unchangeable, but not so for GROUP and WORLD.

If you do not define a protection code for a file when creating it, the system applies a default protection. If a version of the file already exists, protection is taken from the previous version. For a new file, protection is determined in one of the two following ways:

- If the directory where the file is to be catalogued has an associated access control entry that specifies the DEFAULT_PROTECTION entry, then the specified protection is used.
- If the directory does not have the DEFAULT_PROTECTION entry, then the default process protection is used. The default process protection is established explicitly with the SET PROTECTION/DEFAULT command, or by default when you log in.

Show the current default protection by entering the SHOW PROTECTION command as follows:

```
$ SHOW PROTECTION
SYSTEM=RWED, OWNER=RWED, GROUP=RE, WORLD=NO ACCESS
```

This response is the system default protection. It indicates that SYSTEM and OWNER have all types of access, members of the owner's group have READ and EXECUTE access, and all other users (WORLD) have no access.

To determine the current protection associated with a specific file or files, use the /PROTECTION qualifier with the DCL command DIRECTORY, as in the following example:

```
$ DIRECTORY/PROTECTION  MYFILE.REC
Directory DBA1:[CRAMER]

MYFILE.REC;5        (RWED,RWED,RW,R)

Total of 1 file.
```

You can change the default protection applied to files created during an interactive session by using the SET PROTECTION/DEFAULT command. The SET PROTECTION/DEFAULT command indicates that the protection code you specify is to be applied to all files subsequently created during the terminal session or batch job (providing that the files are not subject to other sources of protection).

Note that, to completely protect a file, you must protect both the file itself and the directory in which the file is listed. If you have files that must be protected against unauthorized access, be sure to specify the proper protection both for the directories in which the files are listed and for the files themselves.

2.2.2.3 Directory File Protection

Each directory file has a protection associated with it. The directory protection can override the protection of individual files within the directory. For example, if a directory denies WORLD access, WORLD users cannot look up even those files in the directory that permit WORLD access.

For directory file protection, the following access types are supported:

- READ
- WRITE
- EXECUTE
- DELETE
- CONTROL

Having READ access to a file in a directory means that you have the right to examine, print, or copy that file. If you have READ access to a directory file, you can display the contents of the directory file with the DIRECTORY command. For example, if you have access to the directory [JONES], you can enter the following:

```
$ DIRECTORY [JONES]
```

This command generates a display of the files contained in the [JONES] directory.

If you have READ access, you can access any file listed in the directory, unless the protection on that file denies you access. If the protection applied to the whole directory denies you READ access, then you cannot access even those files in the directory that permit access to users in your group.

If you have WRITE access to a directory file, you have the right to modify or write to that directory file. However, you must have both READ and WRITE access to a directory in order to create files in that directory, to rename files in that directory, or to perform any file operation that involves changes to that directory file.

Note that EXECUTE access has a special meaning when applied to directories. EXECUTE-only access to a directory file allows you access only to files that you can identify by name. It also means that you can access files in the directory that are not protected against users in your category, provided that you do not perform an operation that modifies the directory file. However, you cannot list all the entries in a directory by using wildcards.

For example, assume you have EXECUTE-only access to the [JONES] directory, and you enter the following command:

```
$ DIRECTORY [JONES]
```

The system responds with an error message and does not list the files in the [JONES] directory. However, if you know that the file DATAFILE.DAT resides in the [JONES] directory, you can enter the following command:

```
$ TYPE [JONES]DATAFILE.DAT
```

This causes the system to display the contents of the file. Thus, EXECUTE-only access provides some, but not all, of the operations that READ access provides.

If you have DELETE access to a directory file, you have the right to delete that directory file. You must remove all entries from a directory file before you can delete it. When you create a directory file with the CREATE/DIRECTORY command, you do not, by default, get DELETE access. If you want to be able to delete a directory file, you must use the DCL command SET PROTECTION to explicitly assign DELETE access to the OWNER category.

If you have CONTROL access to a directory file, you have the right to change the characteristics of the directory file.

Remember that, to ensure that your disk files are adequately protected, you must make certain they are properly protected at both the directory and file levels. ACLs allow you to further control access to disk files. For more information on ACLs, see the *VMS Access Control List Editor Manual.*

2.2.2.3.1 UIC Directory Protection

You cannot completely protect a file without applying at least the same protection to the directory in which the file resides. For example, if you deny a user all access to a file but allow that user READ access to the file's directory, the user cannot access the contents of the file but can see that it exists. Conversely, a user allowed access to a file and denied access to the file's directory (or one of the parent directories) cannot see that the file exists.

Note: **To protect sensitive files, the directory protection alone is not adequate. You must also protect each individual file contained within the directory.**

By default, top-level directories receive UIC-based protection (S:RWE,O:RWE,G:RE,W:E) and no ACL. Subdirectories receive UIC-based protection minus any DELETE access or default protection ACEs from the parent directory.

To specify UIC-based protection explicitly when creating a directory, use the /PROTECTION qualifier of the CREATE/DIRECTORY command. You cannot specify an ACL for the directory until the directory is created. To change the UIC-based protection of an existing directory, use the SET PROTECTION command (apply this command to the directory file). To specify or change the ACL of an existing directory, edit the directory file's ACL (see Section 2.2.2.1.1).

You can limit but not prohibit directory access by specifying EXECUTE access but not READ access. EXECUTE access on a directory permits you to examine and read files that you know are contained in the directory (that is, you know the file specifications) but prevents you from displaying a list of the files in the directory.

2.2.2.4 Magnetic Tape File Protection

In general, the protection that applies to a magnetic tape volume also applies to all the files on that volume. For VMS ANSI-labeled magnetic tapes, however, distinct file-level protection is also supported. For VMS ANSI-labeled magnetic tapes, file-level protection can be determined by the contents of the accessibility field, which is located in the first header label of each file.

Each time a VMS ANSI-labeled magnetic tape file is opened for processing, the installation routine examines the contents of the accessibility field of that file. Depending on the value returned by this routine, access to the file in question will either be granted, denied, or defaulted to the VMS protection scheme applied to the volume containing that file (see Section 2.1.3).

2.2.2.5 Protection of Mail Files

Note: **(Requires the Common Utilities Option.)**

Mail files receive the protection (S:RW,O:RW,G,W). Files of type MAI created with the Mail Utility EXTRACT/MAIL command receive the protection (S:RW,O:RW,G,W).

2.2.2.6 Displays of Ownership and Protection

You can display ownership and protection information with the following commands and qualifiers:

Command	Display
DIRECTORY/ACL filespec	File's ACL
DIRECTORY/OWNER_UIC filespec	File's UIC
DIRECTORY/PROTECTION filespec	File's UIC-based protection
DIRECTORY/SECURITY	All of the above
DIRECTORY/FULL filespec	All of the above and more
SHOW ACL obj-name	Device, file, logical name table, or global section's ACL
SHOW PROCESS	Process UIC
SHOW PROTECTION	Default file protection
SHOW DEVICES/FULL device-name	Device UIC and protection

2.3 Device Protection

In general, device protection controls the ability to allocate the device and is specified by granting READ access in an ACL. To specify an ACL for a disk device, use the SET ACL/OBJECT_TYPE=DEVICE command. For example, to grant READ access to the disk device WORKDISK to a user with the alphanumeric UIC [FRED], enter the following:

```
$ SET ACL/OBJECT_TYPE=DEVICE/ACL=(IDENTIFIER=[FRED],ACCESS=READ) -
_$ WORKDISK
```

Note that, when you mount an ACL for a disk device, the ACL is associated with the device not with the disk volume. For example, if you mount a disk device on WORK1 and specify the preceding SET ACL/OBJECT_TYPE=DEVICE command and then dismount the disk device, the ACL protection remains on WORK1 but not on the disk device.

The only protection that applies to a nonfile device is the ability to allocate it, specified by READ access. By default, nonfile devices such as mailboxes are unprotected. Interactive terminals are set up to provide complete access to system users and no access to all other users. (Note that access here refers to access via an application program. The device protection on a terminal does not control who can log in on it.) You can change the protection of a nonfile device through the use of ACLs or by changing the standard UIC protection. Modify the ACL with the DCL command SET ACL/OBJECT_TYPE=DEVICE. Modify the UIC protection with the DCL command SET PROTECTION/DEVICE (requires OPER privilege). For example, the following command allows users holding the PAYROLL identifier to have READ access to TERMINAL3:

```
$ SET ACL/OBJECT_TYPE=DEVICE/ACL=(IDENTIFIER=PAYROLL,ACCESS=READ)-
_$ TERMINAL3
```

3 Preparing Volumes for Private Use

You can prepare private disk and magnetic tape volumes for routine operations by using the DCL commands ALLOCATE, INITIALIZE, and MOUNT. Each of these DCL commands is discussed in the sections that follow.

If you are interested in setting up public volumes, see the description in the *Guide to Setting Up a VMS System.*

3.1 Setting Up a Private Volume

Under some circumstances, it may be desirable to perform your work on a device that cannot be accessed by unauthorized users. By creating a private volume and mounting it on a device allocated exclusively to your process, you can perform your work without fear of interference from other users.

To set up your private volume, perform the following steps:

1 Use the *ALLOCATE* command to assign a disk drive or magnetic tape drive exclusively to your process.

2 Use the *INITIALIZE* command to format the volume and write an identifying label on the volume.

3 Use the *MOUNT* command to make a volume, and the files or data it contains, accessible to your process.

See the section dealing with the individual DCL command for a complete description of each parameter and qualifier. Also see the DCL commands ALLOCATE and INITIALIZE in the *VMS DCL Dictionary,* and for more information on MOUNT, see the *VMS Mount Utility Manual.*

3.2 Allocating Disks and Magnetic Tape Drives to Your Process

Before you can begin processing files or data on a private volume, you must first allocate a drive to your process. Use the DCL command ALLOCATE to logically assign a disk drive or a magnetic tape drive to your process.

Whether you are allocating a disk drive or a magnetic tape drive, the format for the ALLOCATE command is as follows:

```
ALLOCATE device-name[:] [logical-name]
```

Command Parameters

device-name[:]

Specifies the drive on which the volume will be loaded. The device name can be a physical, generic, or logical name. A physical device name consists of a device code, alphabetic controller designation, and a unit number. A generic device name consists only of the device code. A logical name must equate to a physical or generic name. Use of the colon is optional but recommended by DIGITAL.

[logical-name]
Specifies an optional logical name to be associated with the specified disk or magnetic tape drive.

The ALLOCATE command allocates only one device to a process. Except for a list of generic device names, ALLOCATE does not accept lists of device names in the command string. Although you can specify a list of generic devices, the first available device will be the only one to be allocated. For a list of available devices, refer to the Software Product Description (SPD) supplied with your system software.

In the examples that follow, the first two show how to use the ALLOCATE command to allocate disk drives, while the next three show how to allocate magnetic tape drives.

Examples

1

```
$ ALLOCATE DM: DISK
```

In this example, the ALLOCATE command requests that the first available RK06 or RK07 be assigned to your process. The logical name DISK is placed in your process logical name table and assigned the name of the allocated device. Other users are unable to access the device.

2

```
$ ALLOCATE DMB2:
%DCL-I-ALLOC, _MARS$DMB2: allocated
```

In this example, the ALLOCATE command specifies a physical device named DMB2, which requests the allocation of a specific RK06 or RK07 disk drive; that is, unit 2 on controller B. The response from the ALLOCATE command indicates that the device was successfully allocated.

If you want to allocate a particular type of device, use the /GENERIC qualifier with the ALLOCATE command. For example, device DM could be an RK06 or RK07 disk. If you specifically want to allocate an RK07, use the /GENERIC qualifier in the following way:

```
$ ALLOCATE/GENERIC RK07  MYDISK
```

In this case, the system allocates the first available RK07 device to your process. For further discussion of the /GENERIC qualifier as well as all of the qualifiers applicable to the ALLOCATE command, see the *VMS DCL Dictionary.*

3

```
$ ALLOCATE MTA1:
%DCL-I-ALLOC, _MARS$MTA1: allocated
```

In this example, the ALLOCATE command specifies a physical device named MTA1.

The operating system informs you that MTA1 has been allocated.

4

```
$ ALLOCATE MF,MT,MS DRIVE
%DCL-I-ALLOC, _MARS$MTA0: allocated
```

In this example, the ALLOCATE command specifies a list of generic device names. At a minimum, a generic device name consists of the device code; a controller designation is optional. Only one of the specified generic devices is allocated. Each element in the list must represent a unique generic device type.

The VMS operating system informs you that drive MTA0 has been allocated. Although it is not indicated in the message, the system also assigns the logical name DRIVE to the drive MTA0.

5

```
$ ALLOCATE DRIVE1: D1
%DCL-I-ALLOC, _MARS$DBA3: allocated
```

In this example, the ALLOCATE command specifies a logical name, DRIVE1, as the device name and assigns a new logical name, D1. (This example assumes that DRIVE1 has already been defined as the physical device DBA3.)

The VMS operating system informs you that DRIVE1 has been allocated. Although it is not indicated in the message, VMS also assigns the new logical name D1 to the drive DRIVE1.

If you want to allocate a specific type of magnetic tape device, use the /GENERIC qualifier. For example, if you want to allocate a TU78 device specifically, you would use the /GENERIC qualifier with the ALLOCATE command, as follows:

```
$ ALLOCATE/GENERIC TU78  TAPE_TU78
```

In this case, the system would allocate the first available TU78 device to your process. For a further discussion of the /GENERIC qualifier as well as all of the qualifiers applicable to the ALLOCATE command, see the *VMS DCL Dictionary*.

3.3 Initializing a Volume

Before you can write files or data to a disk or magnetic tape volume, the volume must be initialized. You can use the DCL command INITIALIZE to format and write a label to the volume.

The INITIALIZE command does the following:

- Invalidates all existing data on the volume, if any, and creates a new file structure
- Writes a label on the volume to identify it
- Defines the owner UIC and the protection for the volume

Whether you are initializing a disk or magnetic tape volume, the format for the INITIALIZE command is as follows:

INITIALIZE device-name[:] volume-label

Command Parameters

device-name[:]

Specifies the name of the device on which the volume is to be physically mounted and then initialized. To prevent initializing another user's volume, you should allocate a device before you initialize the volume. Prior allocation is not required, however.

volume-label

Specifies the identification to be encoded on the volume. You can specify up to 12 alphanumeric characters for a disk volume, or up to 6 alphanumeric characters for a magnetic tape volume. Alphabetic characters are automatically changed to uppercase. The first character of a disk volume-label specification must be alphanumeric.

There are some cases where you might be prevented from accessing and initializing a particular volume. For example, if the volume that you want to initialize previously contained data, the protection code may prevent you from accessing and initializing that particular volume. In the case of a magnetic tape volume, you may not be able to initialize the volume if the first file on the volume has not reached its expiration date or if the volume or file accessibility is such that the installation routine provided by the VMS operating system does not allow you to access the volume.

If the volume is protected or if the expiration date on the first file has not been reached and you are not the owner or a SYSTEM user, you must have VOLPRO user privilege to override volume protection. If you do not have VOLPRO privilege, you can ask the previous owner of the volume or another user who does have READ/WRITE access (the system manager or operator, for example) to initialize it for you. If the installation routine provided by the VMS operating system (or by a user-designed installation routine) does not allow you access, then consult your system manager.

When you give the volume to another user for initialization, you should specify the following:

- The label you want to have written on the volume
- The protection code and owner UIC you want assigned to the volume

When you obtain a magnetic tape or disk volume, place identification on the outside of the volume so that it can be easily identified.

The next two sections describe how to initialize disk and magnetic tape volumes.

3.3.1 Initializing a Disk Volume

By default, the INITIALIZE command builds a Files–11 structure on your new volume. The default format for disk volumes in the VMS operating system is called the Files–11 On-Disk Structure Level 2. The INITIALIZE command can also initialize disk volumes in the Files–11 On-Disk Structure Level 1 format.

You do not need special privileges to override logical protection on a blank disk volume (that is, a volume that has never been written to) or on a disk volume that is owned by your current UIC or by UIC [0,0]. In all other cases, you must have the user privilege VOLPRO to initialize a disk volume.

The following examples include typical cases of initializing a disk.

Examples

1
```
$ INITIALIZE DISK USER_DISK
```

In this example, the volume is given the label USER_DISK. DISK is the logical name of the device on which it is mounted.

2
```
$ ALLOCATE DJA2:  TEMP
%DCL-I-ALLOC, _MARS$DJA2 allocated
$ INITIALIZE  TEMP:  BACKUP_FILE
```

This example shows how to initialize an RA60 volume. First, the drive is allocated to ensure that no one else can access it. Then, when the volume is physically loaded on the drive, the INITIALIZE command initializes it. Refer to the Software Product Description (SPD) for a complete list of devices that are available.

3.3.2 Initializing a Magnetic Tape Volume

Use the INITIALIZE command to initialize a magnetic tape volume. The default format for magnetic tape volumes in the VMS operating system is based on Level 3 of the ANSI and ISO standards for magnetic tape labels and file structure for informational interchange).

Use the DCL command INITIALIZE to encode VMS ANSI-labeled format on a magnetic tape volume. INITIALIZE writes labels to an empty file on the magnetic tape volume in the following order:

1. A volume label
2. File-header labels with the file sequence number set to 0
3. Two tape marks framing an empty file (BOT and EOT)
4. Corresponding end-of-file labels (EOF)
5. A double tape mark, specifying logical end-of-volume

The following example describes how to initialize a magnetic tape volume:

```
$ INITIALIZE TAPE USER
```

In this example, the magnetic tape volume is given the label USER. TAPE is the logical name of the device on which it is mounted.

Note that, if you use ANSI "a" characters (which are not alphanumeric) on the volume label on magnetic tape, you must enclose the volume name in quotation marks.

3.4 Mounting a Volume

Before you can begin processing files or data on your private disk or magnetic tape volume, make sure the volume is mounted. Use the DCL command MOUNT to make a disk or magnetic tape volume and the files or data it contains accessible to your process.

When you enter the MOUNT command, the system checks the following:

- That the device has not been allocated by another user
- That the device protection allows you to allocate the device
- That a volume is physically loaded on the device specified
- That the label on the volume matches the label specified

You can mount a single volume or a volume set. Binding volumes into a volume set allows you to extend the space available for your files by adding volumes to the set, rather than defining new volumes.

The procedures for creating and mounting disk volume sets and magnetic tape volume sets (as opposed to single volumes) are described in the sections that follow. Whether you are mounting a disk or a magnetic tape volume, the format for entering the MOUNT command is as follows:

```
$ MOUNT device-name[:][,...] [volume-label[,...]] [logical-name[:]]
```

Command Parameters

device-name[:][,...]
Specifies the physical device name or logical name of the device on which the volume is to be mounted. If you specify more than one device name for a disk or magnetic tape volume set, separate the device names with either commas (,) or plus signs (+).

volume-label[,...]
Specifies the label on the volume. If you specify more than one volume label, separate the labels with either commas or plus signs. The volumes must be in the same volume set. For magnetic tape volumes, the labels must be specified in ascending order according to relative volume number.

The volume-label parameter is not required when you mount a volume with the /FOREIGN or /NOLABEL qualifier or when you specify /OVERRIDE=IDENTIFICATION. To specify a logical name when you enter either of these qualifiers, type any alphanumeric characters in the volume-label parameter position.

logical-name[:]
Defines a 1 through 255 alphanumeric character string to be associated with the device.

If you do not specify a logical name for a disk drive, the MOUNT command assigns the default logical name DISK$volume-label to individual disk drives; it assigns the default logical name DISK$volume-set-name to the device on which the root volume of a disk volume set is mounted. Similarly, if you do not specify a logical name for a magnetic tape drive, the MOUNT command assigns only one logical name, TAPE$volume-label, to the first magnetic tape device in the list. For a volume set, no logical name is assigned unless you specify one.

The MOUNT command places the name in the job logical name table, unless you specify /GROUP or /SYSTEM. In the latter cases, it places the logical names in the group or system logical name table. You should avoid assigning a logical name that matches the file name of an executable image in SYS$SYSTEM. Such an assignment will prohibit you from invoking that image.

At many installations, operators perform the physical mounting (and dismounting) of both system and private disk and magnetic tape volumes. Since operators at such installations assist you in your MOUNT requests, you do not need to include the /ASSIST qualifier with the MOUNT command. For example, the following command notifies the operator of your mount request and displays a message at your terminal:

```
$ MOUNT DMA1: DISK VOL1
%MOUNT-I-OPRQST, PLEASE MOUNT DEVICE _MARS$DMA1:
```

After the device has been successfully mounted, you are notified with the following message:

```
%MOUNT-I-MOUNTED, DISK mounted on _DMA1:
```

As an alternative to requesting a specific device, you might want to request a device type. In the following example, MOUNT allocates an available device of the specified type and requests operator assistance in mounting it:

```
$ MOUNT DB: USER_DISK DISK
%MOUNT-I-OPRQST, Please mount volume USER_DISK in device _NODE$DBA0:
%MOUNT-I-MOUNTED, USER_DISK       mounted on _NODE$DBA0:
%MOUNT-I-RQSTDON, operator request canceled - mount completed successfully
```

DISK is the logical name created when the RP06 disk unit is allocated. Since the device is allocated to your process, no other user can access the volume. Your access to USER_DISK is determined by the volume protection code and the volume UIC.

Operator assist messages are sent to all operators enabled to receive TAPE and DISK messages. Thus, if operator assistance is needed for mounting a disk device, a message is sent to disk operators.

Any operator reply to a MOUNT request is written to SYS$OUTPUT to be displayed on the user's terminal or written in a batch job log.

If no operator is available (operator is not enabled) to receive and respond to a MOUNT request, a message is displayed to inform you of the situation. A volume placed in the requested drive needs no additional operator assistance. Note that you can specify the /NOASSIST qualifier to avoid operator assistance.

The following sections provide examples of mounting volumes; however, they do not all include operator assistance messages.

3.4.1 Mounting a Disk Volume

Use the following procedure to allocate, initialize, and mount a disk volume:

```
$ ALLOCATE DMA2:  TEMP
%DCL-I-ALLOC, _MARS$DMA2: allocated
$ INITIALIZE  TEMP:  BACKUP_FILE
$ MOUNT  TEMP:  BACKUP_FILE
%MOUNT-I-MOUNTED, BACKUP_FILE mounted on _DMA2:
```

If you want to mount a foreign disk volume (that is, one having a file structure other than Files–11), use the /FOREIGN qualifier in conjunction with the MOUNT command. The MOUNT/FOREIGN command makes the contents of your volume available to the system but makes no assumptions concerning its file structure. In the following case, assume that DISK is the logical name assigned to the device at the time of disk allocation:

```
$ MOUNT/FOREIGN DISK
%MOUNT-I-MOUNTED, BACKUP_FILE      mounted on DISK$DMA2:
```

Note that MOUNT reports a volume label even though the disk is mounted as a foreign device. MOUNT reports the label because the disk has a Files–11 structure; if a disk does not have a recognized file structure, no label is reported.

You need the user privilege VOLPRO to mount a Files–11 structured disk with the /FOREIGN qualifier, unless its owner UIC matches your own.

You must use the /FOREIGN qualifier if you want to use the VMS Bad Block Locator Utility (BAD) to locate and record bad blocks on your disk volume. BAD is useful for media preparation and is thus distinct from the volume preparation tasks described in this chapter. To invoke BAD, use the DCL command ANALYZE/MEDIA. For more information on BAD, see the description in the *VMS Bad Block Locator Utility Manual*

In addition to the /FOREIGN qualifier, many other qualifiers are supported for the MOUNT command. The /SYSTEM qualifier is described in the *Guide to Setting Up a VMS System*. The /BIND qualifier is discussed in the next section, which describes how to mount disk volume sets. For a complete list of the qualifiers, see the DCL MOUNT command.

3.4.2 Mounting a Disk Volume Set

When you mount a disk volume set, the volume label specified in the list must correspond to a device name in the same position in the device name list.

Two or more disk volumes can be bound into a volume set. The first volume in the set is called the *root volume*. Each volume in the set is identified by a volume number relative to the root volume, which is always relative to volume 1.

A disk volume set has a single directory structure. The master file directory (MFD) for the entire volume set resides on the root volume, which is always the first volume in the set.

When a disk volume set is on line and mounted, all files and directories in the set can be accessed by specifying either of the following:

- Device name of the device on which the root volume is mounted

- Logical name assigned to the volume set when it was mounted

Use the /BIND qualifier to create a disk volume set. When you use the MOUNT command with the /BIND qualifier to create a disk volume set, the /BIND qualifier identifies a volume set by assigning it a volume set name that applies to all volumes in the set. It also identifies the root volume and creates the directory structure for the volume.

When you create files on a volume set, the file system allocates space for the files anywhere on the set, wherever the most space exists. When existing files on any volume are extended, extension occurs on the same volume, unless the volume is physically full. You can add new volumes to a volume set whenever additional space is needed.

For example, all disk volumes that are mounted on a daily basis can be bound into a volume set. Since this set contains all user file directories, users do not need to specify device names in file specifications to access files on any volume in the volume set. In fact, the physical location of a file is of no concern to users of the system.

Note: **Do not bind your system disk into a volume set. Volume sets are not supported by VMS software updates and optional product installation. If certain system files move or extend to other volumes in the set, the system may fail to boot.**

No special privileges are required to use volume sets. However, you must have WRITE access to the index file on all volumes you are attempting to bind into a volume set; this usually means you also must have a system UIC, have the user privilege SYSPRV, or be the owner of the volumes.

You can create a disk volume set from newly initialized volumes, or you can create a volume set by extending an existing volume that already contains a directory structure and files. You can also add volumes to an existing volume set. The next three sections contain examples of how to create and mount each type of disk volume set.

3.4.2.1 Creating a Disk Volume Set from New Volumes

The following steps show how to create a disk volume set from new disk volumes. This example assumes there are no files or data on the volumes to be bound.

1 Allocate the necessary devices and physically load the volumes.

2 Initialize each volume in the set by entering the following:

```
$ INITIALIZE DB1:  PAYVOL1
$ INITIALIZE DB2:  PAYVOL2
$ INITIALIZE DB3:  PAYVOL3
```

When you initialize volumes for a volume set, you can also use other qualifiers with the INITIALIZE command to define the volume ownership and protection. Protection and ownership information is obtained from the root (first) volume. The protection and ownership of the other volumes is ignored.

3 Enter the following MOUNT/BIND command to create the volume set:

```
$ MOUNT/BIND=MASTER_SET -
_$ DB1:, DB2:, DB3:   PAYVOL1, PAYVOL2, PAYVOL3
```

This MOUNT/BIND command defines the volume set name, MASTER_SET, and defines the relative volume numbers of the volumes PAYVOL1, PAYVOL2, and PAYVOL3.

A disk volume set name can have from 1 to 12 alphanumeric characters. The volume set name must be different from all volume labels within the set, and all labels in the set must be unique.

The order of the device names corresponds to the volume labels specified: PAYVOL1 must be physically loaded on DB1, PAYVOL2 on DB2, and PAYVOL3 on DB3.

PAYVOL1, which is listed first in the list of labels, becomes the root volume of the set. The master file directory (MFD) for PAYVOL1 contains the directory structure for the entire volume set.

Note that the MOUNT/BIND command creates the volume set and mounts the volumes. When this command completes successfully, all volumes in the set are ready for use — user file directories can be created.

The /BIND qualifier must be used only once to create the volume set. In subsequent use, the volume set may be mounted with a single MOUNT command. The following example illustrates the use of one MOUNT command to mount a previously created volume set:

```
$ MOUNT DB1,DB2,DB3 PAYVOL1,PAYVOL2,PAYVOL3
```

3.4.2.2 Creating a Disk Volume Set from an Existing Volume

The following example shows how to create a disk volume set from an existing volume, assuming that the volume USERFILES already contains a directory structure and files and that the volume is currently located on the device DM1:

```
$ DISMOUNT/NOUNLOAD DM1:

$ INITIALIZE DM2: USERFILES2
$ MOUNT/BIND=USERS -
_$ DM1:, DM2:    USERFILES, USERFILES2
```

The initial volume USERFILES must be specified first; it becomes the root volume of the set. When you create a volume set from an existing volume, you must specify that volume first because the file system must build on the existing directory structure.

Note that, if you attempt to create a volume set from two or more volumes that already contain files and data, the file system does not issue an error message when you enter the MOUNT/BIND command. However, the volumes are unusable as a volume set because the directory structures are not properly bound.

3.4.2.3 Adding Volumes to a Disk Volume Set

This section describes how to add volumes to an existing volume set. The following example assumes that the volume set named MASTER_SET is on line and mounted and has volumes named PAYVOL1, PAYVOL2, and PAYVOL3:

```
$ INITIALIZE DB4: PAYVOL4
$ MOUNT/BIND=MASTER_SET DB4: PAYVOL4
```

The MOUNT command binds the volume PAYVOL4 with the existing volume set and makes the volume ready and available for use. Note that, if the volume set MASTER_PAY was mounted with the /SYSTEM, /GROUP, or /SHARE qualifier, the MOUNT/BIND command that adds a volume to the set must also specify the appropriate qualifier.

When you add a volume to an existing set, the only volume in the set that must be mounted is the root volume, relative volume 1. None of the other volumes need be mounted.

You can also add a volume to a set at the same time you mount the set. The following procedure assumes an existing volume set named MASTER_SET with volumes named PAYVOL1, PAYVOL2, and PAYVOL3:

```
$ INITIALIZE DB4:  PAYVOL4
$ MOUNT/BIND=MASTER_SET  DB1:, DB2:, DB3:, DB4:
_$PAYVOL1, PAYVOL2, PAYVOL3, PAYVOL4/SYSTEM
```

Note that the first device/volume pair listed in the MOUNT/BIND command is the root volume of the set. When you add a volume to a set while mounting the set, you must list the root volume first.

Note: Once a volume is bound into a volume set, it cannot be "unbound."

You can add volumes to an existing volume set at any time. The maximum number of volumes in a set is 255.

3.4.3 Mounting a Magnetic Tape Volume

When mounting a magnetic tape volume, specify, along with the MOUNT command, the following: any qualifiers you choose; a device name; and, optionally, a label and logical name.

For a discussion of how to mount magnetic tape volume sets, see Section 3.4.4.

The next two sections describe procedures and commands for mounting single-volume, ANSI-labeled, magnetic tapes.

3.4.3.1 Mounting an ANSI-Labeled Volume

When you use the MOUNT command to mount a magnetic tape volume, VMS software checks to see whether the volume has a VMS or a non-VMS ANSI-labeled format. If the format is ANSI-labeled, the following are checked:

- The volume identifier field
- The protection on the ANSI-labeled volume as described in Section 2.1.3.

Mount an ANSI-labeled volume by including the device name and volume identifier as follows (specifying a logical name is optional):

```
$ MOUNT  MT:  ELAINE ET
%MOUNT-I-OPRQST, please mount volume ELAINE in device $MTA1:
%MOUNT-I-MOUNTED, ELAINE mounted on MTA1:
%MOUNT-I-RQSTDON, operator request canceled -- mount completed successfully
```

MOUNT finds an available MT drive, MTA1, and requests operator assistance. The message displayed at the user terminal indicates which drive has been selected. At this point, you (or the operator) load the magnetic tape on the drive, and the mount operation completes. No operator response is necessary. The display informs you that the volume named ELAINE is mounted on the drive MTA1. Although not indicated in the message, MOUNT also assigns the logical name ET to the volume ELAINE.

When used with the MOUNT command, the qualifiers described in the next section affect the label format of a volume or of the magnetic tape file system used to process an ANSI-labeled volume, or both. Unless otherwise noted, you must have VOLPRO privilege to use any of these qualifiers when the volume is a VMS ANSI-labeled volume containing protection that restricts your process from accessing the volume.

3.4.3.2 Using MOUNT Command Qualifiers

This section describes some of the command qualifiers you can use when mounting a magnetic tape volume. For a complete list of all the command qualifiers supported for the mounting of magnetic tape volumes, see the *VMS Mount Utility Manual.*

/BLOCKSIZE=n

Use the /BLOCKSIZE=n qualifier to specify the block size for the magnetic tape. The range of valid values for *n* varies and depends on the density of the volume, whether the data is for input or output, and whether the operation uses RMS.

By default, VMS writes 2048-byte blocks, which conforms to the ANSI standard. Although the VMS operating system allows you to specify a block size larger than 2048 bytes, a larger block size does not conform to ANSI standards.

You must specify /BLOCKSIZE when mounting volumes that do not support the second file header label as defined in the ANSI standard or that support a block size smaller than 2048 bytes. For example, you must specify /BLOCKSIZE=512 to mount an RT–11 volume.

If you want to write files on a magnetic tape volume, use the /BLOCKSIZE qualifier to set the block size to other than 2048. If you want only to read files from a magnetic tape volume, you do not have to use the /BLOCKSIZE qualifier, since VMS will automatically handle the block size given by the second file header label.

The minimum blocksize for ANSI standard magnetic tapes is 18, while the maximum is 2048. The minimum blocksize for VMS magnetic tapes is 14, while the maximum is 65,532.

/FOREIGN

The /FOREIGN qualifier should be used when a magnetic tape volume is not in standard ANSI or ISO format or when a disk volume is not in Files–11 format.

/OVERRIDE

The /OVERRIDE=(option[,...]) qualifier inhibits one or more of the access checks performed by MOUNT and the magnetic tape file system. The options are as follows:

- ACCESSIBILITY — If the installation allows, this option will override any character in the accessibility field of the volume and file header labels. The necessity of this qualifier is defined by the installation. That is, each installation has the option of specifying a routine that the magnetic tape file system will use to process this field. By default, the VMS operating system provides a routine that checks this field in the following manner:
  - If the magnetic tape was created on a version of the VMS operating system that conforms to Version 3 of ANSI, then this option must be used to override any character other than an ASCII space.
  - If a VMS protection is specified and the magnetic tape conforms to an ANSI standard that is later than Version 3, then this option must be used to override any character other than an ASCII 1.

  For more information on accessibility-based protection for VMS ANSI-labeled magnetic tapes, see Chapter 2.
- EXPIRATION — Overrides the expiration dates of a volume and its files. Use this qualifier when the expiration date (in the first file-header label) of any file that you want to overwrite has not been reached.
- IDENTIFICATION — Overrides the volume identifier in the volume label. Use this qualifier to mount a volume for which you do not specify the volume identifier. Only the volume identifier field will be overridden. Volume protection, if any, is preserved.
- OWNER_IDENTIFIER — Overrides the processing of the owner identifier field. You use this option when you need to interchange protected magnetic tapes between VMS and other DIGITAL operating systems.
- SETID — Prevents MOUNT from checking the file-set identifier in the first file-header label of the first file on a continuation volume. Use this qualifier only for ANSI-labeled volumes on which the file-set identifier of the first file on a continuation volume differs from the file-set identifier of the first file of the first volume that was mounted.

/OWNER_UIC=uic

The /OWNER_UIC=uic qualifier overrides the UIC written in the second volume label and assigns the UIC you specify while the volume is mounted. For magnetic tape volume sets in which a continuation volume is written, the UIC specified at mount time is written to the volume only if the /PROTECTION qualifier was specified either at mount time or when the volume was initialized. This does not change the protection on any volumes already created.

You can specify the UIC variable in the format:

[g,m]

g is either an octal number in the range 0 through 37776 that denotes the group number or an alphanumeric value (consisting of 1 through 31 characters) that describes the group.

m is either an octal number in the range 0 through 177776 that denotes the member number or an alphanumeric value (consisting of 1 through 31 characters) that describes the member.

Either square ([ ]) or angle (<>) brackets are required in the UIC specification. For more details on UIC-based protection, see Chapter 2.

/CACHE=TAPE_DATA

The /CACHE qualifier with the TAPE_DATA option enables the write cache for a tape device if the tape controller supports one. /NOCACHE is the default for mounting tape devices. You must specify TAPE_DATA to enable the write cache. If the tape controller does not support a write cache, the option is ignored. This lets you specify /CACHE=TAPE_DATA in command procedures that work with a variety of devices and controllers.

This option enables a form of controller-based, write-back caching. The write-back caching feature significantly improves the overall performance of streaming tape drives. Under some rare failure conditions, however, some written data can be lost in the cache of the controller. If a failure occurs, the magnetic tape being written becomes seriously flawed or unreadable, and you must repeat the entire process used to write to the tape. When a failure occurs, you are always notified with an error message.

/PROTECTION=code

The /PROTECTION=code qualifier overrides the VMS protection written in the second volume label and assigns the protection code you specify to the volume while it is mounted. For magnetic tape volume sets in which a continuation volume is written, the protection code you specify will be written to the continuation volume. By default, your process UIC also will be written to the continuation volume unless you explicitly specify an alternate UIC with the /OWNER_UIC qualifier described above.

Valid protection codes include READ and WRITE access for GROUP and WORLD users; EXECUTE and DELETE access are not applicable to magnetic tape volumes. SYSTEM users and the volume owner always have READ and WRITE access to magnetic tape volumes regardless of the protection code that you specify. Section 2.2.2 describes access and protection codes.

/HDR3

The /NOHDR3 qualifier controls whether special VMS header labels are written to a volume. Privilege is not required for this qualifier. The default is /HDR3, which allows VMS header labels to be written to a volume. When the /NOHDR3 qualifier is used, long VMS file names are truncated to 17 characters. Use the /NOHDR3 qualifier when writing to volumes that will be read by a system other than VMS, such as the RT–11 system, which does not process all file-header labels correctly.

/RECORDSIZE=n

The /RECORDSIZE=n qualifier specifies the number of bytes in each record. This qualifier does not require privilege. Use this qualifier when you mount volumes without the second file-header label (such as RT–11 volumes), or when you mount volumes with the /FOREIGN qualifier, to provide RMS with the size of fixed-length records or the maximum size of variable-length records.

The record size must be less than or equal to the specified or default block size. Refer to the /BLOCKSIZE qualifier (described previously) for details. The VMS operating system does not write records smaller than 14 bytes on output. However, the VMS Convert Utility, described in the *VMS Convert and Convert/Reclaim Utility Manual,* allows you to pad and extend the size of records up to and greater than the 14-byte minimum.

Two other qualifiers that are important for mounting magnetic tape volumes are /AUTOMATIC and /INITIALIZE. These qualifiers are described in the following sections.

3.4.4 Mounting a Magnetic Tape Volume Set

When mounting a magnetic tape volume set, begin by following the procedures described in Section 3.4.3 for mounting a single magnetic tape volume. The number of volume identifiers need not equal the number of device names specified. Thus, when you mount a magnetic tape volume set, you can specify more volume identifiers than device names or more device names than volumes.

The number of devices you specify directly affects the action taken by the magnetic tape file system when processing continuation volumes in a volume set. For example, when the number of devices is greater than the number of volumes, the magnetic tape files system requests a continuation volume to be mounted on the first drive from the list that does not have a volume mounted.

The next two sections describe how to create and mount a magnetic tape volume set. The manner in which continuation volumes are handled by the magnetic tape file system is also described.

3.4.4.1 Creating a Magnetic Tape Volume Set

If you do not create a volume set explicitly, the VMS system creates one when necessary. If you have not mounted a volume set and a continuation volume is required, the magnetic tape file system requests that a continuation volume be mounted and implicitly creates a volume set. For example, if the magnetic tape file system encounters an EOT mark while writing a volume, it sends a message to the operator console requesting that another volume be mounted.

After you mount the next volume, the magnetic tape file system writes the volume and header labels and then reissues the pending write requests to the continuation volume. The file-set identifier in the first file-header label of all files written to the continuation volume is the file-set identifier of the first file on the first volume. The file-set identifier for VMS volume sets is always that of the first file of the first volume that was mounted in the set.

To explicitly create a volume set with three volumes, follow this procedure:

1 Allocate a drive on which you will load each volume by entering the following ALLOCATE commands:

```
$ ALLOCATE MTA0:
%DCL-I-ALLOC, _MARS$MTA0: allocated

$ ALLOCATE MTA1:
%DCL-I-ALLOC, _MARS$MTA1: allocated

$ ALLOCATE MTA2:
%DCL-I-ALLOC, _MARS$MTA2: allocated
```

2 Initialize the volumes. You should specify the density and the access protection in addition to the device name and the volume identifier in the INITIALIZE commands, as in the following command lines:

```
$ INITIALIZE/DENSITY=1600/PROTECTION=(G:RW) MTA0: TAPE1
$ INITIALIZE/DENSITY=1600/PROTECTION=(G:RW) MTA1: TAPE2
$ INITIALIZE/DENSITY=1600/PROTECTION=(G:RW) MTA2: TAPE3
```

3 Mount the volumes by entering the following MOUNT command. You should include the device name and volume identifier. Specifying a logical name for the volume set is optional.

```
$ MOUNT MTA0:,MTA1:,MTA2: TAPE1,TAPE2,TAPE3 TEST
%MOUNT-I-MOUNTED, TAPE1 mounted on _MTA0:
%MOUNT-I-MOUNTED, TAPE2 mounted on _MTA1:
%MOUNT-I-MOUNTED, TAPE3 mounted on _MTA2:
```

The system not only confirms which volumes have been mounted but also indicates on which drive each volume has been mounted.

The system mounts and verifies only the volumes that are physically loaded on the drives at mount time. However, the volume identifiers of additional volumes that you specify are not verified until the volumes are accessed.

You can check the densities, volume labels, UICs, and relative volume numbers of the volumes that are mounted on drives. To do so, specify the SHOW DEVICES/FULL command. If you specify a generic device code for the magnetic tape drives, such as MT, information for all the drives of that type configured in the system is displayed. To display information for a volume mounted on a specific drive, specify the physical device code, consisting of the generic device code, the controller designation, and the unit number followed by a colon. For more information on the SHOW DEVICES command, including examples of displays returned by the SHOW DEVICE /FULL command, see Chapter 4 and the *VMS DCL Dictionary.*

3.4.4.2 Mounting Continuation Volumes in a Volume Set

When mounting a magnetic tape volume set, follow the general procedures described in the previous section for creating a magnetic tape volume set. Once the volume set has been created, however, there is no need to initialize the volumes in the set when you mount the volume set.

You need not allocate a drive for each volume in the volume set. The magnetic tape file system requests that volumes be switched to appropriate drives when continuation volumes are required.

The VMS operating system stores, but cannot verify, the volume identifiers of volumes you specify but do not physically mount on drives at mount time. VMS later verifies the volume identifiers of such volumes when the volumes are accessed.

The VMS operating system supports the continuous processing of mounted volumes in a magnetic tape volume set through automatic volume switching. To do this, the magnetic tape file system uses automatic volume recognition (AVR) and automatic volume labeling (AVL).

To take advantage of this automatic volume switching capability, you must have more than one magnetic tape drive allocated to your volume set. If you have two or more magnetic tape drives allocated to a volume set, the magnetic tape file system switches volumes for you automatically by sequentially selecting the next magnetic tape drive allocated to the volume set. The magnetic tape file system expects the next volume in the volume set to be loaded on that drive.

If the file system is writing to the volume set, it creates a label for the magnetic tape and initializes the magnetic tape with that label and with the protection characteristics set for the first of the volume set. If the magnetic tape file system is reading the volume set, it generates the label and tries to mount the next magnetic tape with that label. If the drive has no magnetic tape loaded on it, or the wrong magnetic tape, the magnetic tape file system sends a message to the operator console notifying the operator to either mount a magnetic tape or mount the correct magnetic tape.

Before processing continuation volumes, the magnetic tape file system processes the protection on that volume (as described in Section 2.1.3). If the magnetic tape file system determines that the user does not have access to the volume, then a message is sent to the operator to take some action.

The label generated fills the six-character volume identifier field. The first four characters of the field contain the first four characters of the label specified for the previous volume in the volume set. (If the label is less than four characters, the volume identifier field is padded with underscores.) The fifth and sixth characters contain the relative volume number for that reel in the volume set. Note that this allows VMS to generate only 99 unique labels for a given volume set.

With automatic volume switching enabled, the operator can load a magnetic tape on the next drive allocated to the magnetic tape volume set anytime before the volume being processed reaches the EOT mark. The magnetic tape file system mounts and initializes (if INITIALIZE was specified originally) the next magnetic tape in the volume set and then notifies the operator that the switch has occurred.

In the following example, the volume with the identifier TAPE is mounted on MTA0:

```
$ MOUNT MTA0:,MTA1:,MTA2:  TAPE
```

Continuation volumes for this set should be loaded on the magnetic tape drives in the following order: MTA1, MTA2, MTA0, MTA1, MTA2, and so forth.

To explicitly override automatic volume switching, specify the /NOAUTOMATIC qualifier when mounting a magnetic tape volume. The default is /AUTOMATIC. If you allocate only one drive to the magnetic tape volume set, you implicitly disable automatic volume switching.

To ensure that any volume added to the magnetic tape volume set will be initialized prior to being written to, mount the volume with the /INITIALIZE=CONTINUATION qualifier. The default is /NOINITIALIZE.

The next example demonstrates the use of the /INITIALIZE=CONTINUATION qualifier for mounting volume sets. It also shows how volume identifiers are generated for continuation volumes.

```
$ INITIALIZE MTA0: MAIN
$ MOUNT/OVERRIDE=IDENTIFIER/INITIALIZE=CONTINUATION MTA0:,MTA1:
```

The volume labeled MAIN is mounted on MTA0. The second volume in the set receives the volume identifier MAIN02 and is mounted on MTA1. The third volume in the set receives the volume identifier MAIN03 and is mounted on MTA0. In the following example, the first volume in the set is labeled SUN and is mounted on MTA0. The second volume receives the identifier SUN_02 and is mounted on MTA1. The third volume receives the identifier SUN_03 and is mounted on MTA0:

```
$ MOUNT MTA0:,MTA1: SUN
```

The next example illustrates a continuation volume with two volume identifiers.

```
$ MOUNT MTA0:,MTA1: SUN,MOON
```

In this case, SUN and MOON are mounted on MTA0 and MTA1 respectively. If a third volume is added to the set, it will be given the identifier MOON03 and be mounted on MTA0.

3.5 Dismounting a Volume

When you have finished processing the files or data on your disk or magnetic tape volume, you can use the DISMOUNT command to explicitly dismount a single volume or an entire volume set.

If you explicitly dismount a single volume in a volume set, VMS dismounts all the volumes in the set. For disk volume sets, however, it is possible to explicitly dismount a single volume in the volume set without dismounting the entire set. To do this, you must use the /UNIT qualifier.

When you enter the DCL command DISMOUNT, the volume is automatically unloaded from the drive. You can override this automatic unloading of your volume by specifying the /NOUNLOAD qualifier with the DISMOUNT command.

Even when you specify the /NOUNLOAD qualifier with the DISMOUNT command, your volume is still logically dismounted from the drive; however, the volume remains physically loaded on the drive. If you use the /NOUNLOAD qualifier to dismount a magnetic tape volume, the volume remains loaded on the magnetic tape drive and the magnetic tape reel is rewound to the BOT mark.

If you plan to remount or reinitialize a volume you are dismounting, you can save time and eliminate unnecessary handling of that volume by using the /NOUNLOAD qualifier with the DISMOUNT command.

The following examples show how to use the DISMOUNT command. The first example employs the /NOUNLOAD qualifier.

```
$ DISMOUNT/NOUNLOAD MTA1:
$
```

In this example, the magnetic tape volume is logically dismounted and remains loaded on drive MTA1. Also, the magnetic tape reel is rewound to the BOT mark. The VMS operating system returns you to DCL level.

The DISMOUNT command is also used to dismount foreign volumes. The following command dismounts a volume that had been mounted with the /FOREIGN qualifier on DBA0:

```
$ DISMOUNT DBA0:
$
```

In this example, the volume that had been mounted with the /FOREIGN qualifier on DBA0 is dismounted and automatically unloaded. The VMS system returns you to DCL level.

As mentioned previously, use the DISMOUNT command to dismount an entire volume set. If you explicitly dismount any volume in a disk or magnetic tape volume set, the entire volume set is dismounted. For example, if you had a volume set that consisted of DBA3 and DBA4 and you entered the following command, the entire volume set would be dismounted:

```
$ DISMOUNT DBA3:
```

You should always explicitly dismount a volume or volume set with the DISMOUNT command, or with a command procedure containing that command, before physically unloading that volume.

A volume is dismounted and unloaded automatically if you log out of the job from which you had mounted the volume. If the system fails, however, the drive is not automatically dismounted.

Note that data corruption can occur if a volume has not been explicitly dismounted and the system fails. For magnetic tape volumes, data corruption can occur if you unload a volume that contains an open file for which file-trailer labels have not been written. When you remount the volume and attempt to access the file without file-trailer labels, you receive the following error message:

```
%MTAACP-magnetic tape position lost
```

You will be able to access all the files (on that magnetic tape volume) preceding the file whose file-trailer labels had not been written. However, you will not be able to access the file without file-trailer labels.

Note that the dismount of a volume is done by the file system and is not complete until all the open files on the volume have been closed. Thus, a substantial amount of time can pass between the time you enter the DISMOUNT command and the completion of the dismount. Always wait for the drive to unload before you remove the volume. (You can verify that the dismount is complete by entering the DCL command SHOW DEVICES.)

If the device you are dismounting was allocated with an ALLOCATE command, it remains allocated after it is dismounted with the DISMOUNT command. If the device was implicitly allocated by the MOUNT command, the DISMOUNT command deallocates it.

For more information on the DISMOUNT command, see the *VMS DCL Dictionary*.

3.6 Deallocating Drives

The process of allocation reserves a device for exclusive use by your process. The device remains allocated to your process until you explicitly deallocate it or until you log out from your process. Once you have allocated the device, other users cannot access that device until you explicitly deallocate it or log out.

Use the DCL command DEALLOCATE to explicitly deallocate a disk drive or magnetic tape drive that has been allocated to your process. A complement to the ALLOCATE command, the DEALLOCATE command logically disconnects a drive from your process and returns it to the pool of devices.

The following example shows how to explicitly deallocate a magnetic tape drive or a disk drive:

```
$ DEALLOCATE MTA1:
$
```

In this example, the DEALLOCATE command logically disconnects magnetic tape drive MTA1 from your process. The VMS operating system returns you to DCL level.

Since logging out of a process from which drives have been allocated automatically deallocates all explicitly and implicitly allocated drives, you do not have to explicitly deallocate a disk or magnetic tape drive that has been allocated to your process. However, it is a good practice to use the DEALLOCATE command (or a command procedure containing this command) to explicitly deallocate all the drives you allocated with the ALLOCATE command.

3.7 Using Command Procedures to Set Up Volumes

Since private disk and magnetic tape volumes frequently must be set up before you can perform operations on them, you may want to design command procedures to facilitate the set-up procedures. This section contains examples of command procedures that can be used to set up disk and magnetic tape volumes for routine operations.

You can tailor command procedures to meet the needs of your own set-up tasks. The command procedure examples in this section, although general in nature, can serve as guiding strategies for you.

The next two sections contain examples of command procedures designed to set up disk and magnetic tape volumes for routine processing.

3.7.1 Designing Command Procedures to Set Up Disk Volumes

The following command procedure is designed to allocate, initialize, and mount a disk volume. You can use a text editor, such as EDT or EVE, to create a file to contain your command procedure. Assume that a file named SETUP.COM has been created and that it contains a very basic command procedure, which, when executed, allocates and mounts a disk. Construct this command procedure by entering the following text:

```
$ ! Place a disk in the drive
$ IF P1 .EQS. "" THEN INQUIRE P1 "enter device name"
$ IF P2 .EQS. "" THEN INQUIRE P2 "enter volume label"
$ IF P3 .EQS. "" THEN INQUIRE P3 "enter logical name"
$ ALLOCATE 'P1'
$ MOUNT 'P1' 'P2' 'P3'
```

This command procedure, although very simple, accomplishes the task of allocating and mounting a disk each time it is executed. It is designed to prompt you for the device name, volume label, and logical name of the disk device that you want to allocate and mount. By assigning logical names to your disks, you can use this command procedure to allocate and mount devices over and over again.

You can take further advantage of the power of a command procedure by including a few additional tasks as well. For example, you could design the SETUP.COM command procedure to deallocate and dismount the disk. The command procedure example used to set up a magnetic tape (described in the next section) takes advantage of some of these options.

To execute the SETUP.COM command procedure, enter the following command:

```
@SETUP
```

You can also write command procedures to mount a volume from a batch job. By using logical names to refer to devices and files, you can use the same command procedures without modification each time you want to access a volume.

For example, if you use the same RK07 disk pack to back up your files on a weekly basis, you can submit as a batch job a command procedure such as the following:

```
$ MOUNT  DM:  BACK_UP_GMB  RK
$ BACKUP/REPLACE  *.*  RK:*.*
$ DIRECTORY/FULL/OUTPUT=BACKUP.LOG  RK:
$ DISMOUNT RK:
```

In this command procedure, the MOUNT command finds and allocates a device of the type DM and creates a logical name RK. The MOUNT command substitutes the equivalence name in the message displayed at the operator console.

When the MOUNT command notifies the operator to mount the correct volume, the job waits until the operator responds. When the operation is completed, the DISMOUNT command dismounts the device RK.

3.7.2 Designing Command Procedures to Set Up Magnetic Tape Volumes

The command procedure below, which is more sophisticated and detailed than the previous example, is designed to set up a magnetic tape for processing. The ALLOCATE and MOUNT/FOREIGN commands are included in this command procedure. Using a text editor, construct the command procedure in the following way:

```
$ ! First mount the tape on the drive
$ ON CONTROL_Y THEN GOTO EXIT
$ ON ERROR THEN GOTO EXIT
$ WRITE SYS$OUTPUT "Welcome to FETCH."
$ WRITE SYS$OUTPUT " "

$ L1:  INQUIRE/NOPUNC PHYS "Have you placed the volume in the drive? "
$ IF .NOT. PHYS THEN GOTO L1
$ INQUIRE/NOPUNC DRIVE "Which drive is the volume mounted on? "
$ DRIVE = DRIVE - ":"
$ ALLOCATE 'DRIVE'
$ MOUNT/FOREIGN 'DRIVE'
$ ON ERROR THEN GOTO COMMAND_LOOP
$ !
$ COMMAND_LOOP:  INQUIRE/NOPUNC OPTION "FETCH> "

$ IF OPTION .EQS. "DIR" THEN GOTO DIR
$ IF OPTION .EQS. "EXIT" THEN GOTO EXIT
$ IF OPTION .EQS. "FETCH" THEN GOTO FETCH
$ IF OPTION .EQS. "HELP" THEN GOTO HELP
$ IF OPTION .EQS. "LIST" THEN GOTO LIST
$ GOTO COMMAND_LOOP
$ !
$ DIR:  INQUIRE SPEC "Filespec"
$ DIR 'SPEC'
$ GOTO COMMAND_LOOP
$ HELP:
$ WRITE SYS$OUTPUT "Enter any of the following commands at the prompt:"
$ WRITE SYS$OUTPUT " "
$ WRITE SYS$OUTPUT " "
$ WRITE SYS$OUTPUT "DIR        (To search for a file)"
$ WRITE SYS$OUTPUT " "
$ WRITE SYS$OUTPUT "EXIT       (To exit this program)"
$ WRITE SYS$OUTPUT " "
$ WRITE SYS$OUTPUT "FETCH      (To perform a BACKUP RESTORE operation)"
$ WRITE SYS$OUTPUT " "
$ WRITE SYS$OUTPUT "HELP       (To read this text)"
$ WRITE SYS$OUTPUT " "
$ WRITE SYS$OUTPUT "LIST       (To perform a BACKUP LIST operation)"
$ GOTO COMMAND_LOOP
$ !
$ FETCH:  INQUIRE FILE "Filespec"
$ INQUIRE SAVESET "Save set name"
$ LINE := BACKUP/LOG 'DRIVE':'SAVESET'/SELECT='FILE'
$ INQUIRE EXCLUDE "Enter any filespecs you want excluded"
$ IF EXCLUDE .EQS. "" THEN GOTO L2
$ LINE := 'LINE'/EXCLUDE=('EXCLUDE')
$ !
$ L2:  INQUIRE/NOPUNC TO "Where do you want the file(s)? (RET for current directory)"
$ IF TO .EQS. "" THEN GOTO REPLACE
$ LINE := 'LINE' 'TO'
$ GOTO L3
$ REPLACE:  LINE := 'LINE' []
$ !
$ L3:  INQUIRE/NOPUNC NEW "Create a new version if file already exists? "
$ IF .NOT. NEW THEN GOTO NOT
$ LINE := 'LINE'/NEW_VERSION
$ !
$ NOT:  LINE := 'LINE'/OWNER_UIC=ORIGINAL
$ LINE
$ GOTO COMMAND_LOOP
$ !
$ LIST:  INQUIRE SPEC "Filespec"
$ INQUIRE SAVESET "Save set name"
$ INQUIRE/NOPUNC OUTPUT "What do you want to call the list file? (RET for SYS$OUTPUT ")
$ IF OUTPUT .EQS. "" THEN GOTO NOOUT
$ LINE := BACKUP/LIST='OUTPUT' 'DRIVE':'SAVESET'/SELECT=('SPEC')
$ GOTO L4
$ NOOUT:  LINE := BACKUP/LIST 'DRIVE':'SAVESET'/SELECT=('SPEC')
$ !
$ L4:  INQUIRE EXCLUDE "Enter any filespecs you want excluded"
$ IF EXCLUDE .EQS. "" THEN GOT L5
$ LINE := 'LINE'/EXCLUDE=('EXCLUDE')
$ !
$ L5:  LINE
$ GOTO COMMAND_LOOP
$ !
$ EXIT:
$ DISMOUNT 'DRIVE'
$ DEALLOCATE 'DRIVE'
```

Assume this command procedure is contained in a file named FETCH.COM; you would execute this command procedure by entering the following:

```
@FETCH
```

This command procedure is more complex than the two previous ones, which were both used to set up a disk. As with the former command procedures, the procedure contained in FETCH.COM also accomplishes the basic task of allocating the device and mounting the volume.

In addition to its allocating and mounting functions, the command procedure contained in FETCH.COM is designed to prompt you for input. For example, it specifically asks you if the magnetic tape is on the drive. Also, note that this command procedure is designed to do a BACKUP restore operation. It prompts you for specific options on the restore operation.

Finally, this command procedure explicitly dismounts your magnetic tape volume and deallocates the drive after your task has completed.

4 Manipulating Files

You can manipulate disk and magnetic tape files by using the DIGITAL Command Language (DCL). In particular, you can use DCL commands to perform the following tasks:

- Retrieve disk and magnetic tape file information
- Modify disk and magnetic tape file characteristics
- Access files residing on disk and magnetic tape volumes

DCL commands enable you to manipulate files in the following ways:

- SHOW commands retrieve disk and magnetic tape file information, such as device and protection characteristics, and display this information on your terminal.
- SET commands modify disk and magnetic tape file characteristics, such as protection or UIC information.
- The DCL command language can be used to access disk and magnetic tape files for read or write operations.

In addition to manipulating files through DCL, you can write user programs to assist you in routine file-manipulation tasks. You can write these programs in either VAX MACRO or in one of the higher-level languages supported by the VMS operating system. If you want to manipulate individual records within files — that is, to access files at the record level — you should write programs that include RMS facilities. Examples of the RMS facilities used to manipulate files at the record level are included in the *VMS Record Management Services Manual*.

You can use DCL to manipulate disk and magnetic tape files at the file level. Note the following restrictions on the use of DCL commands for manipulating files on disk and magnetic tape volumes:

- The SUBMIT command cannot access files on allocated devices. You can submit files on private disk volumes if you mount the volume as a shareable volume. To submit a file on a magnetic tape volume, you first must copy the file to a shared disk volume and then submit it from the disk.
- You can print a file from a privately owned volume. Note, however, that the volume containing the file you wish to print must remain mounted until after the file has completed printing. If you do not want to wait until the printing completes, you can copy the file directly to the line printer, using the DCL command COPY as in the following example:

  ```
  $ COPY MYPHILE.DAT LPAO:
  ```
- Most DCL commands require file-structured devices. For a list of those commands that do not require file-structured devices, see the *VMS DCL Dictionary*.

- You can execute a command procedure that resides on a magnetic tape volume as long as the procedure does not invoke other procedures and does not issue any GOTO commands that refer to labels in the procedure preceding the GOTO command. In this case, it would be better to copy the command procedure from the magnetic tape volume to a local disk from which you can then invoke the command procedure.

Note that you cannot use DCL commands to read or write files that are not in the standard formats supported by VMS (these formats are described in greater detail in the *Guide to Using VMS Command Procedures*).

4.1 Using DCL to Retrieve File Information

The DCL command language provides commands that enable you to retrieve information about disk and magnetic tape files, volumes, and devices. You can use the following DCL commands to retrieve such information:

- DIRECTORY
- SHOW DEVICES
- SHOW MAGTAPE
- SHOW ACL
- SHOW PROTECTION
- SHOW QUOTA

See the *VMS DCL Dictionary* for a complete list of the command qualifiers and parameters applicable to each of these DCL commands.

4.1.1 Retrieving Directory Information

Use the DCL command DIRECTORY to retrieve information about a file or a group of files residing on a disk or magnetic tape volume.

You can use the DIRECTORY command to list the names of all of the files in a particular directory, or you can use it in conjunction with a file specification to list the names of specific files in a given directory. When you include certain command qualifiers along with the DIRECTORY command, you can retrieve information in addition to the names of the files. See the *VMS DCL Dictionary* for a list of qualifiers that can be used with the DIRECTORY command.

The following examples illustrate three cases of retrieving information from the [MALCOLM] directory, which resides on a disk with the logical name DISK$DOCUMENT.

Examples

1

```
$ DIRECTORY AVERAGE.*

Directory DISK$DOCUMENT:[MALCOLM]

AVERAGE.EXE;6      AVERAGE.FOR;6      AVERAGE.LIS;4     AVERAGE.OBJ;12

Total of 4 files.
```

2

```
$ DIRECTORY/SIZE=USED/DATE=CREATED/VERSIONS=1/PROTECTION  AVERAGE

Directory DISK$DOCUMENT:[MALCOLM]

AVERAGE.EXE;6       6        10-APR-1988 15:43 (RWED,RWED,RWED,RE)
AVERAGE.FOR;6       2         2-APR-1988 10:29 (RWED,RWED,RWED,RE)
AVERAGE.LIS;4       5         9-APR-1988 16:27 (RWED,RWED,RWED,RE)
AVERAGE.OBJ;6       2         9-APR-1988 16:27 (RWED,RWED,RWED,RE)

Total of 4 files, 15 blocks.
```

3

```
$ DIRECTORY/FULL/VERSIONS=1   [MALCOLM...]AVERAGE.EXE

Directory DISK$DOCUMENT:[MALCOLM]

AVERAGE.EXE;6                 File ID:  (4098,149,0)
Size:           36/36         Owner:    [DOCUMENTATION,MALCOLM]
Created:  27-JUN-1988 12:22:26.30
Revised:  27-JUN-1988 12:22:51.35 (2)
Expires:   <None specified>
Backup:    3-JUL-1988 22:03.09
File organization:  Sequential
File attributes:    Allocation: 36, Extend: 36, Global buffer count: 0
                    No version limit
Record format:      Variable length, maximum 255 bytes
Record attributes:  Carriage return carriage control
Journaling Enabled : None
File protection:    System:RWED, Owner:RWED, Group:RE, World:
Access Cntrl List:  None

Total of 1 file, 36/36 blocks.

Directory DISK$DOCUMENT:[MALCOLM.TEST]

AVERAGE.EXE;1                 File ID:  (7714,29,0)
Size:           36/36         Owner:    [DOCUMENTATION,MALCOLM]
Created:  15-APR-1988 10:12
Revised:  15-APR-1988 10:12 (1)
Expires:   <None specified>
Backup:   15-APR-1988 22:41
File organization:  Sequential
File attributes:    Allocation: 36, Extend: 36, Global buffer count: 0
                    No version limit
Record format:      Variable length, maximum 255 bytes
Record attributes:  Carriage return carriage control
Journaling Enabled : None
File protection:    System:RWED, Owner:RWED, Group:RE, World:
Access Cntrl List:  None

Total of 1 file, 36/36 blocks.

Grand total of 2 directories, 2 files, 72/72 blocks.
```

Directory structures do not apply to magnetic tape volumes. However, you can use the DIRECTORY command to search for files on magnetic tape volumes. This use of the DIRECTORY command is also discussed in Section 4.3.2, which describes how to access magnetic tape files for read and write operations.

To find the names of all files on a magnetic tape volume mounted on MTA2, enter the following command:

```
$ DIRECTORY MTA2:
```

This directory command lists the file names and file types of all files on the magnetic tape.

As in the case of the disk examples included above, you can use wildcard characters in directory specifications for magnetic tapes, as in the following command:

```
$ DIRECTORY MFA0:*.*;*
```

In response to this command, the VMS operating system searches the entire volume set and returns both ANSI and VMS file names. (The difference between these two types of magnetic tape files is described in Section 4.3.2).

4.1.2 Retrieving Device Information

Use the DCL command SHOW DEVICES to retrieve information about the availability of devices on your system.

When you enter the SHOW DEVICES command without specifying a device or using a qualifier, information about all devices on the system is displayed. If you specify a device name, SHOW DEVICES displays information about that device. If you use certain qualifiers with SHOW DEVICES, information is displayed about those devices that currently have volumes mounted or that have been allocated to processes. See the *VMS DCL Dictionary* for a list of qualifiers that can be used with the SHOW DEVICES command.

The device name displayed by the system uses the format *ddcu,* where *dd* is the device code, *c* is the control, and *u* is the unit. If the system is part of a VAXcluster environment, the device name includes the node name in the format *node$ddcu;* where, *node* refers to the node name of the system that the device resides on, *dd* refers to the device type, *c* refers to the controller designation, and *u* refers to the unit number. See the *VMS VAXcluster Manual* for a complete description of the device name format on clusters.

The following examples illustrate three instances of how the SHOW DEVICES command can be used.

Examples

1

```
$ SHOW DEVICES

Device                Device     Error   Volume          Free   Trans  Mnt
 Name                 Status     Count    Label         Blocks  Count  Cnt
DBA0:                 Mounted        0  VMS              47088    115    1
DBA1:                 Mounted        0  USERPACK1        45216      2    1
DBA2:                 Mounted        3  DOCUMENT          8068     20    1
DBA5:                 Mounted        0  MASTERP          28668      1    1
DBA6:                 Online         0
DBA7:                 Mounted        0  PROJECT         110547      1    1
DMA0:                 Online         0
DLA0:                 Online         0
DYA0:                 Online         0
DYA1:                 Online         0
DRA3:                 Mounted        0  RES26APR         29317      1    1
MFA0:                 Online         8
MFA1:                 Online         1
MTA0:                 Mounted        9  BACKUP             453      1    1
MTA1:                 Online         0
```

The SHOW DEVICES command displays the following information for each device on the system:

- Device name.
- Device status and characteristics. (Status indicates whether the device is on line; characteristics indicate whether the device is allocated, spooled, or has a volume mounted on it and if the volume is mounted foreign.)
- Error count.
- Volume labels.
- Number of free blocks on the volume (disk only).
- Transaction count.
- Number of mount requests issued for the volume.

2

```
$ SHOW DEVICES/FULL DMA0

Disk $1$DMA0: (NODE1), device type RK07, is online, allocated,
      served to the cluster, error logging enabled.

    Error count                 2    Operations completed              4527
    Owner process         "SMITH"    Owner UIC                        [0,0]
    Owner process ID     24400133    Dev Prot     S:RWED,O:RWED,G:RWED,W:RWED
    Reference count             1    Default buffer size                512
    Allocation class            1
```

This SHOW DEVICES command requests a full listing of the status of the RK07 device DMA0. The device is located on NODE1 in a VAXcluster.

3

```
$ SHOW DEVICES/FULL NODE2$

Disk $1$DBA0: (NODE2), device type RP05, is online, mounted, file-oriented
    device, shareable, served to cluster via MSCP Server, error logging is
    enabled.

    Error count                 0    Operations completed              120
    Owner process              ""    Owner UIC                     [303,2]
    Owner process ID     00000000    Dev Prot     S:RWED,O:RWED,G:RWED,W:RWED
    Reference count             1    Default buffer size               512
    Total blocks           171798    Sectors per track                  22
    Total cylinders           411    Tracks per cylinder                19
    Allocation class            1

    Volume label       "HIGHNOON"    Relative volume number              0
    Cluster size                3    Transaction count                   1
    Free blocks             94425    Maximum files allowed           21474
    Extend quantity             5    Mount count                         8
    Mount status           System    Cache name         "_$255$DUA8:XQPCACHE"
    Extent cache size          64    Maximum blocks in extent cache   9442
    File ID cache size         64    Blocks currently in extent cache    0
    Quota cache size            0    Maximum buffers in FCP cache      421

  Volume status:  subject to mount verification, file high-water marking, write-
      through caching enabled.
  Volume is also mounted on NODE1, NODE4, NODE3.

Disk NODE2$DBC1:, device type RP06, is online, error logging enabled.

    Error count                 0    Operations completed                0
    Owner process              ""    Owner UIC                       [0,0]
    Owner process ID     00000000    Dev Prot     S:RWED,O:RWED,G:RWED,W:RWED
    Reference count             0    Default buffer size               512
    Host name             "NODE2"    Host type, available        V780, yes

Disk NODE2$DMA0:, device type RK07, is online, error logging enabled.

    Error count                 0    Operations completed                0
    Owner process              ""    Owner UIC                       [0,0]
    Owner process ID     00000000    Dev Prot     S:RWED,O:RWED,G:RWED,W:RWED
    Reference count             0    Default buffer size               512
    Host name             "NODE2"    Host type, available        V780, yes
      .
      .
      .
```

The command line SHOW DEVICES/FULL NODE2$ produces a full display of information about each device on NODE2 on the VAXcluster system. Information is shown here only for the first three devices: a mounted device and two that are not mounted.

4.1.3 Retrieving Magnetic Tape Device Information

You can use the DCL command SHOW MAGTAPE to display the current characteristics and status of a specified magnetic tape device.

You can enter the SHOW DEVICES command to find available magnetic drives on your system. The SHOW MAGTAPE or SHOW DEVICE /FULL commands enable you to retrieve additional information about the characteristics of a particular magnetic tape device.

When you enter the SHOW MAGTAPE command at your terminal, you receive the following prompt:

```
_Device:
```

You must then specify the name of the magnetic tape device for which you want to display the characteristics and status.

The following example illustrates how the SHOW MAGTAPE command is used to retrieve information about MTA0:

Example

```
$ SHOW MAGTAPE MTA0:
MTA0:,          device type TU77, is online, error logging is enabled

Error count            0    Operations completed              0
Owner process        " "    Owner UIC                     [0,0]
Owner process ID 0000000    Dev Prot S:RWED, O:RWED, G:RWED, W:RWED
Reference Count        0    Default buffer size            2048
Density              800    Format                    Normal-11
Volume Status: no-unload on dismount, odd parity)
```

This SHOW MAGTAPE command displays the characteristics of the device MTA0. Among other characteristics, it displays the device type, density, and format.

4.1.4 Retrieving Disk File Protection Information

The DCL command SHOW PROTECTION displays the current process default protection. This protection is applied to files created during your terminal session or to batch jobs, where defaults from directories or previously existing versions are not available.

You can change the default protection at any time with the SET PROTECTION command. (The SET ACL and SET PROTECTION commands are discussed in Section 4.3, which describes how to modify file characteristics.)

This section is not applicable to magnetic tapes. Although you can use the SHOW PROTECTION or SET PROTECTION commands to show or set the default protection of magnetic tapes, the protection is not written to the magnetic tape volume unless you specify the /PROTECTION= qualifier with the INITIALIZE command when you are initializing the magnetic tape volume. See the description of initializing magnetic tape volumes in Chapter 3.

The next example illustrates how the SHOW PROTECTION command can be used in conjunction with the SET PROTECTION command to display and modify the protection characteristics of a disk file.

Example

```
$ SHOW PROTECTION
SYSTEM=RWED, OWNER=RWED, GROUP=RE, WORLD=NO ACCESS
$ SET PROTECTION=(GROUP:RWED,WORLD:RE)/DEFAULT
$ SHOW PROTECTION
SYSTEM=RWED, OWNER=RWED, GROUP=RWED, WORLD=RE
```

This SHOW PROTECTION command requests a display of the current protection defaults. The SET PROTECTION/DEFAULT command is then used to change the file access allowed to other users in the same group as well as to miscellaneous system users.

4.1.5 Retrieving Disk Quota Information

Frequently, it is important to limit the amount of disk space certain users consume. The VMS Disk Quota Utility (DISKQUOTA) gives the system manager this capability. Users who have READ access to the quota file can enter the SHOW QUOTA command to determine how much disk space any user on the system has been allocated. Users who do not have READ access to the quota file can use the SHOW QUOTA command to determine their own allotments.

You can manage the checking on a per-volume basis at mount time by using the MOUNT qualifier /[NO]QUOTA. (You can also use DISKQUOTA to allow or disallow amounts of disk space.) You must have the VOLPRO user privilege, or your UIC must match the UIC written on the volume, in order to use the MOUNT command qualifier /QUOTA.

Enter the DCL command SHOW QUOTA to find out whether a quota exists for any specific user on a specific disk. The display that results from the SHOW QUOTA command gives the quotas used, authorized, and available.

A user can have a quota for any disk volume on the system. In some cases, the user is permitted a certain authorized limit plus an overdraft limit. Generally, only the authorized limit applies. However, certain system programs, such as editors, can employ the overdraft feature when the authorized limit is exceeded.

If you run out of disk space during the creation of a file, you receive a system message. If you cannot obtain sufficient space by purging or deleting unnecessary files, you may need to contact the system manager to increase your disk quota. If you attempt to write a file to a spooled printer, you must have WRITE access and have sufficient quota on the disk associated with that printer.

Examples

1

```
$ SHOW QUOTA
User [DOCUMENTATION,MALCOLM] has 2780 blocks used, 7220 available,
of 10000 authorized and permitted overdraft of 500 blocks on DISK$
```

The SHOW QUOTA command displays the amount of disk space authorized, used, and still available on the current default disk for the present user. The permitted overdraft in this example is 500 blocks.

2

```
$ SHOW QUOTA /USER=[DOCUMENTATION,JONES]/DISK=XXX1:
%SYSTEM-F-NODISKQUOTA, no disk quota entry for this UIC
```

This SHOW QUOTA command displays the fact that the user with UIC [DOCUMENTATION,JONES] has no disk quota allocation on device XXX1.

3

```
$ SHOW QUOTA /USER=[DOCUMENTATION,ELAINE]
User [DOCUMENTATION,ELAINE] has 27305 blocks used, 2305 OVERDRAWN,
of 25000 authorized and permitted overdraft of 4000 blocks on DISK$
```

This SHOW QUOTA command illustrates a user with an overdrawn quota.

4.2 Using DCL to Modify File Characteristics

DCL provides commands that enable you to establish and modify the characteristics of files residing on disk and magnetic tape volumes. The following commands establish or modify file characteristics:

- SET ACL
- SET DIRECTORY (disk only)
- SET FILE (disk only)
- SET MAGTAPE (magnetic tape only)
- SET PROTECTION (disk only)
- SET VOLUME (disk only)

Each of these commands is discussed in detail in the *Guide to Using VMS Command Procedures*. For a complete list of the command qualifiers and parameters applicable to each of these DCL commands, see the *VMS DCL Dictionary*.

4.2.1 Modifying Directory Characteristics

The DCL command SET DIRECTORY modifies the characteristics of one or more directories. The following examples illustrate two uses of the SET DIRECTORY command.

Examples

1

```
$ SET DIRECTORY/VERSION_LIMIT=5/CONFIRM [SMITH.FORTRAN]
```

In this example, the SET DIRECTORY command limits the number of versions to five for files created after the command is issued. The /CONFIRM qualifier requests that you confirm whether or not the specified directory should actually be modified.

2

```
$ SET DIRECTORY/OWNER_UIC=[DOCUMENTATION,GRAY] [DAVIDSON],[USERS]
```

Here, the SET DIRECTORY command modifies both the [DAVIDSON] and [USERS] directories, changing their owner UICs. Use of the /OWNER_UIC qualifier requires SYSPRV or GRPPRV for changing the ownership at the system or group level.

4.2.2 Modifying Disk File Characteristics

Use the DCL command SET FILE to modify the characteristics of one or more files. The examples that follow illustrate three ways the SET FILE command can be used to modify file characteristics.

Examples

1

```
$ SET FILE/EXPIRATION_DATE=15-APR-1988:11:00 BATCH.COM;3
```

This SET FILE command requests that the expiration date of the file BATCH.COM;3 be set to 11:00 A.M., April 15, 1988.

2

```
$ SET FILE/BEFORE=15-APR-88/ERASE_ON_DELETE PERSONNEL*.SAL
```

This SET FILE command calls for all files that match the file specification PERSONNEL*.SAL and that are dated before April 15, 1988. Disk locations are erased for files that are deleted with commands such as DELETE or PURGE.

3

```
$ SET FILE/OWNER_UIC=[DOCUMENTATION,GRAY]/VERSION_LIMIT=100 MYFILE.DAT
```

This SET FILE command modifies the characteristics of the file MYFILE.DAT, changing the owner UIC and assigning a file version limit of 100. Note that the /OWNER_UIC qualifier requires SYSPRV or GRPPRV for changing the ownership at the system or group level.

4.2.3 Modifying Magnetic Tape Device Characteristics

Use the DCL command SET MAGTAPE to define the default characteristics associated with a specific magnetic tape device for subsequent file operations. The device must not be currently allocated to any other user.

The following examples illustrate uses of the SET MAGTAPE command in conjunction with the MOUNT command.

Examples

1

```
$ MOUNT MTB1:/FOREIGN
$ SET MAGTAPE MTB1: /DENSITY=800
```

The MOUNT command mounts a foreign tape on the device MTB1. The SET MAGTAPE command defines the density at 800 bpi for writing to the magnetic tape. (The density is reset only if the magnetic tape has never been written before.)

2

```
$ MOUNT MTA0:/FOREIGN
$ SET MAGTAPE MTA0:/SKIP=FILES:4
```

The MOUNT command mounts a foreign magnetic tape on the device MTA0; the SET MAGTAPE command directs the I/O subsystem to position the magnetic tape to skip four files.

3

```
$ MOUNT MTA1:/FOREIGN
$ SET MAGTAPE/REWIND MTA1:
```

The MOUNT command mounts a foreign tape on the device MTA1; the SET MAGTAPE rewinds the volume.

4.2.4 Modifying File Protection Characteristics

Use the SET PROTECTION command to change or reset the protection characteristics for one or more files. If you include a protection code, the file protection is changed to that specified in the code. When you omit the protection code and do not use the /PROTECTION qualifier, the file protection changes to the default file access established by the SET PROTECTION/DEFAULT command. See the SET PROTECTION/DEFAULT command for information on how to change the default file protection.

All disk and magnetic tape volumes have protection codes that restrict access to the volume. The protection codes for magnetic tape volumes are assigned with the INITIALIZE and MOUNT commands. Protection characteristics on magnetic tape volumes cannot be changed by the SET PROTECTION command.

For disk volumes, each file on the volume can have a different protection associated with it. The SET PROTECTION command and other file-manipulating commands allow you to define the protection for individual files.

If you omit both the code and the /PROTECTION file qualifier, your current default protection (established by the SET PROTECTION/DEFAULT command) is applied to the file.

The examples that follow illustrate using the SET PROTECTION command.

Examples

```
1   $ DELETE INCOME.DAT;3
    %DELETE-W-FILNOTDEL, error deleting DISK1:[SMITH]INCOME.DAT;3
    -RMS-E-PRV, insufficient privilege or file protection violation
    $ SET PROTECTION=OWNER:D INCOME.DAT;3
    $ DELETE INCOME.DAT;3
```

The file INCOME.DAT;3 has been protected against deletion. This SET PROTECTION command changes only the owner's DELETE access for the file INCOME.DAT;3. Now the file can be deleted.

```
2   $ SET PROTECTION -
    _$PAYROLL.LIS/PROTECTION=(SYSTEM:R,OWNER:RWED,GROUP:RW),-
    _$PAYROLL.OUT/PROTECTION=(SYSTEM:RWED,GROUP:RWED,WORLD)
```

In this example, the SET PROTECTION command changes the protection codes applied to two files. To the file PAYROLL.LIS, it gives the system READ access, the owner READ, WRITE, EXECUTE, and DELETE access, and users in the owner's group READ/WRITE access. To the file PAYROLL.OUT, it gives the system and group all types of access; the current access for owner does not change, but world is denied all types of access.

3
```
$ SET PROTECTION A.DAT, B.DAT/PROTECTION=OWNER:RWED, C.DAT
```

The SET PROTECTION command specifies that the file A.DAT should receive the default protection established for the owner's files. The existing protection for the file B.DAT is overridden, only for the OWNER category, to provide READ, WRITE, EXECUTE, and DELETE access. Note that no protection is specified for the file C.DAT at either the command or file level. Thus, like A.DAT, C.DAT receives the default protection.

Since no version numbers are specified in this example, the protection settings affect only the highest versions of the three files.

4
```
$ SET PROTECTION=OWNER:D -
_$[MALCOLM.SUB1]SUB2.DIR/PROTECTION=GROUP:D
```

This SET PROTECTION command changes the protection for the owner and group categories of the subdirectory [MALCOLM.SUB1.SUB2] to permit deletion. However, the protection for world and system categories is not changed.

4.2.5 Modifying User Identification Code Characteristics

Although DIGITAL discourages it, you can use the DCL command SET UIC to establish a new user identification code (UIC) as your default. To use this command, however, you need a special privilege called Change Mode to Kernel (CMKRNL).

If you have the appropriate privilege, you can use the SET UIC command to gain access to a restricted file — that is, a file contained in a directory whose protection restricts access to the owner of that directory. You can also use the SET UIC command to gain access to a restricted magnetic tape volume.

The following examples illustrate how to use the SET UIC command.

Examples

1
```
$ SET UIC [370,10]
```

This command establishes your UIC as [370,10]. You can now read or modify any files whose access is restricted to this UIC.

2
```
$ SET UIC [214,4]
$ SET DEFAULT [ANDERSON]
```

The SET UIC command sets your UIC to [214,4]; the SET DEFAULT command sets the default directory name to [ANDERSON].

3
```
$ SET UIC [ELAINE]
```

This example sets the UIC to be that of the user named ELAINE. You can optionally include the group name in the specification. For example, you could set the UIC to [ET,ELAINE]. Note that, since the user name is unique accross the system, the group name need not be included in the SET UIC command description. However, the user's group name as well as the user name will always be included in UIC displays.

4.2.6 Modifying Volume Characteristics

Use the DCL command SET VOLUME to modify the characteristics of one or more mounted Files–11 disk volumes. In order to use this command, you must have WRITE access to the index file on the volume. If you are not the owner of the volume, you must have either a system UIC or the user privilege SYSPRV. You must then specify the name of one or more mounted Files–11 volumes.

The examples that follow illustrate how the SET VOLUME command can be used.

Examples

1

```
$ SET VOLUME/DATA_CHECK=(READ,WRITE) DBC5
```

This command requests that data checks be performed following all read and write operations to DBC5.

2

```
$ SET VOLUME/LABEL=LICENSES DBC5:
```

This command encodes the label LICENSES on the volume DBC5. Note that, if characters in labels are entered in lowercase, they are changed to uppercase by the /LABEL qualifier.

4.3 Using DCL to Access Files

In addition to executing user programs that perform I/O operations on files residing on disk and magnetic tape volumes, you can use DCL commands to access disk and magnetic tape files for read and write operations.

Note that this section describes how to use DCL commands to access disk and magnetic tape files at the file level (as opposed to the record level). Although DCL does allow you to manipulate files at the record level, performance considerations usually warrant the use of a conventional programming language. DIGITAL recommends that you write programs using the VMS Record Management Services (RMS) facilities that are specifically designed to access files at the record level. You can write these programs in VAX MACRO or in any of the higher-level languages supported by the VMS operating system.

If you wish to access disk and magnetic tape files at the file level, you can take advantage of DCL commands. As mentioned above, you cannot use DCL commands to read or write files that are not in the standard formats supported by the VMS operating system. If the file formats are not standard, the volumes on which they reside must be mounted with the /FOREIGN qualifier.

The following sections provide examples of the steps you can use to access files on disk and magnetic tape volumes. In particular, these sections describe how to manipulate disk and magnetic tape files (at the file level) for READ and WRITE access.

4.3.1 Accessing Disk Files for Read and Write Operations

You can access disk files for both read and write operations. The next three sections show how to use DCL to access disk files for both types of operations.

These sections contain examples of how to do the following:

- Read files from a mounted disk volume
- Write files to a disk volume that must first be allocated, initialized, and mounted
- Write files from a default directory to a volume that must first be allocated, initialized, and mounted on a magnetic tape device

Although the examples used in the following sections show how to access disk files on RK06/RK07 disk packs, they also apply to other devices.

4.3.1.1 Reading Files from a Disk Volume

To read the contents of a disk file, use the DCL command TYPE, which displays the contents of a file on your terminal. To find the exact location of the disk file you want to read, use the DCL command DIRECTORY.

For example, if you wish to read the contents of a file named HISFILE, which is located somewhere in the directory [CHARLES] on a disk device whose logical name is DISK$DOCUMENT, look for the exact location of HISFILE by entering the following command:

```
$ DIRECTORY DISK$DOCUMENT:[CHARLES...]HISFILE.*
```

This command instructs the VMS operating system to search the entire [CHARLES] directory, including all the subdirectories, for all file types and version numbers of HISFILE. The following information will be displayed on your terminal:

```
Directory DISK$DOCUMENT:[CHARLES.MEMO]

HISFILE.UPD;1

Total of 1 file.
```

This display informs you that there is only one version of HISFILE, that its file type is UPD, and that it resides in the [CHARLES.MEMO] directory.

To read the contents of this file, enter the following command:

```
$ TYPE [CHARLES.MEMO]HISFILE.UPD
```

The contents of HISFILE will be displayed on your terminal.

4.3.1.2 Writing Files to a Disk Volume

Before you can write files to a disk volume, the volume must be properly prepared (see Chapter 3).

Because disks are random-access devices and since files must be listed in directories, you must create a directory to contain your files on the disk volume after you have initialized it, as in the following example:

```
$ CREATE/DIRECTORY DMA3:[PUBS]
$ DEFINE P DMA3:[PUBS]
$ COPY *.*  P
$ COPY [PRIMER]*.*  P
$ COPY [COMMANDS]*.*  P
```

The CREATE/DIRECTORY command creates a directory file named [PUBS] on the device DMA3, and the DEFINE command defines the logical name P as DMA3:[PUBS]. The COPY command copies the highest versions of all files in the current default directory and in the directories [PRIMER] and [COMMANDS] to the newly created directory.

4.3.1.3 Writing Files from Disk Volumes to Magnetic Tape Volumes

The next example describes how to use DCL to write files from a default directory on a disk volume to an ANSI-labeled magnetic tape volume. It includes examples that show how to allocate, initialize, and use a magnetic tape to copy a set of your disk files. The procedures are similar to those for copying files from disk volumes to disk volumes. One main difference, however, is that magnetic tapes are sequential-access devices and do not have directories. (The characteristics of magnetic tape files are described more thoroughly in the next section.)

First, allocate a drive as follows:

```
$ ALLOCATE MT:  TAPE_DEVICE
%DCL-I-ALLOC _MARS$MTA2: allocated
```

This ALLOCATE command requests the allocation of a magnetic tape drive whose name begins with MT. TAPE_DEVICE is a logical name, which in this case refers to MTA2.

The system response indicates that unit 2 on controller A was available and is now allocated to you. You can now physically load the magnetic tape on the drive. Be sure the write ring on the magnetic tape is in place; if it is not, you cannot write to the magnetic tape.

Next, initialize the magnetic tape by entering the following:

```
$ INITIALIZE  TAPE_DEVICE:  GMB001 -
_$ /PROTECTION=(GROUP:R,WORLD)
```

The INITIALIZE command specifies the logical name for the volume (TAPE_DEVICE, which in this case refers to MTA2) and the volume label for the magnetic tape volume (GMB001). The label can be no longer than six characters. The /PROTECTION qualifier defines a protection code restricting GROUP access to read and allowing no WORLD access. You can now enter the MOUNT command to mount the volume and write files to it, as in the following example:

```
$ MOUNT  TAPE_DEVICE:  GMB001
%MOUNT-I-MOUNTED, GMB001 mounted on _MTA2:
$ COPY *.*  TAPE_DEVICE:
```

The MOUNT command specifies the device name and volume label of the volume on the device. The COPY command copies the highest versions of all files in your default directory onto the magnetic tape. The file names, file types, and version numbers of the output files default to the same file names, file types, and version numbers as the input files.

If you enter the COPY command with the /LOG qualifier, the system will send a message to the current SYS$OUTPUT device after each file has been copied. You can also use the DIRECTORY command to verify that the files were successfully copied.

```
$ DIRECTORY TAPE_DEVICE:
```

This DIRECTORY command lists the file names and file types of all files on the magnetic tape.

When you have finished using the magnetic tape, dismount and deallocate it as follows:

```
$ DISMOUNT TAPE_DEVICE:
$ DEALLOCATE TAPE_DEVICE:
```

If you do not dismount and deallocate the magnetic tape, the system does so automatically when you log out.

4.3.2 Accessing Magnetic Tape Files for Read and Write Operations

When you request access to an ANSI-labeled volume or a file, the VMS operating system checks at the volume and file level to ensure that your process can access the volume or file. The level at which the system checks access depends on the operation you request and the type of access the operation requires.

When you access a volume or a file, the VMS software reads the volume- and file-header labels to determine whether access to the volume or file is restricted. Which label is read depends on the operation requested. For example, if you want to mount a volume, your process must have access to it. Thus, the operating system reads the volume labels only.

This section describes file-level access for magnetic tapes. For more detailed information on volume-level access and protection requirements for magnetic tapes, see Chapter 2.

Your access to a particular file is determined by the protection that has been set on that file. The expiration date field in the header can prevent you from overwriting or appending to a file immediately preceding the one in question. If the expiration date field has not been reached, the file has not expired. You cannot overwrite an unexpired file unless you specify the /OVERRIDE=EXPIRATION qualifier when you mount the volume.

The manner in which the file is accessed to perform an operation is called the access type. READ or WRITE access, or both, are required.

Before accessing a particular file for a read or write operation, you may want to search the magnetic tape volume for that file. The following section describes how to use the DIRECTORY command to locate a file or group of files on a magnetic tape volume.

4.3.2.1 Locating ANSI-Labeled Magnetic Tape Files for READ or WRITE Access

When you specify a VMS or ANSI file name for a file residing on magnetic tape, the magnetic tape file system compares the file name with the file header labels of each file until it finds a match in the file identifier field of the file header labels.

If you supply a version number in the file name, it is compared with the generation number and generation version-number fields in the first file header label. If you do not specify a version number, the magnetic tape file system neither defaults a version number nor checks the generation number and generation version-number fields. The magnetic tape file system selects the first file on the magnetic tape whose file name in the file identifier field matches the specified file name.

Neither the directory nor the latest version number concept is supported by the VMS operating system for magnetic tape volumes. VMS does not search for or list the latest version of a specified file. The magnetic tape file system cannot increment version numbers of files written to magnetic tape; therefore, two or more files in the same volume set can have the same file name and version number.

Because the magnetic tape file system selects the first matching file name and version number (if specified), the position of the magnetic tape within the volume set determines which file is returned on a search operation. A search operation begins at the current position, so you may want to rewind the volume set before accessing a file.

The search for a matching file and version number (if specified) continues at the beginning of the header-label set of the next file. The search ends when the magnetic tape is positioned at the file where the search began. If the requested file is not found on the current volume, the remaining volumes in the volume set are searched sequentially, according to their relative volume numbers, until either a file name match occurs or the entire volume set is searched.

If a file name match occurs, the internal file identification number is constructed from the file section number, file sequence number, and relative volume number. Although you can access a disk file by its file identification number, you cannot access a magnetic tape file this way.

The VMS operating system does support the use of wildcard characters in file specifications for magnetic tape volumes. However, there are certain restrictions on using wildcard characters with magnetic tape files. Unlike files on disk volumes, which support the asterisk (*), percent sign (%), ellipsis (. . .), and minus sign (-) characters, magnetic tape volumes support only the asterisk and percent-sign wildcard characters with VMS file names. ANSI file names support only the asterisk wildcard character.

The asterisk wildcard character matches file specifications by field or portion of a field. The percent sign wildcard character matches any character in a file specification only by character positions within a field.

With VMS file names, you can specify the asterisk and the percent sign anywhere in the file name and file type field to match file name specifications by field or character position within a field. You cannot use the percent sign in the version number field. Only the asterisk wildcard character can be used in the version number field.

With ANSI file names, a single asterisk in a field is the only wildcard character that can be used. ANSI file names do support the special set of ASCII "a" characters. Unlike VMS file names, which can consist of up to 39 characters each for the file name and file type, ANSI file names can have a maximum of 17 characters in length. Whether you choose VMS file names or ANSI file names depends on the type of applications you want to perform.

The examples that follow illustrate how to use wildcard characters in file specifications to search for files on magnetic tape volumes. These examples also show how the DIRECTORY command can be used in conjunction with magnetic tapes. Note that the DIRECTORY command does not work the same with magnetic tape files as with disk files.

Examples

1

```
$ DIRECTORY MFA1:*.*;*
```

This command instructs VMS to search a volume set. Because asterisks are used in the file specification and the asterisk is a valid wildcard character for both ANSI and VMS file names, both VMS and ANSI file names will be returned. Note that ANSI file names will be returned within quotation mark characters.

2

```
$ DIRECTORY MTA1:%*.*;*
$ DIRECTORY MTA0:*.%*;*
```

In these two commands, the search can only match with VMS file names because the percent sign is not valid for ANSI file names. In the second command, the file type field must contain at least one character. Files with no file type are not returned.

3

```
$ DIRECTORY MTA0:*.;*
```

In this example, the DIRECTORY command instructs VMS to search for files with ANSI file names, as well as VMS file names that have a null file type.

4.3.2.2 Reading Files on Magnetic Tape Volumes

When a magnetic tape file is accessed for a read operation, the magnetic tape is positioned at the beginning of the file section after the file header labels.

You can use the DCL command TYPE to read a file or group of files on the magnetic tape volume and to display the contents of the file on your terminal. For example, if you want to read the contents of a file named TESTFILE.DOC;1 (which you know from your above directory searches is a VMS file residing on the magnetic tape device MTA1), enter the following command:

```
$ TYPE MTA1:TEST*.%*;*
```

You will then receive the following display on your terminal:

```
MTA1:TESTFILE.DOC;1

This is a test file.
```

When a file residing on a magnetic tape volume is accessed only for reading the attributes in the header labels (rather than the data in the file section), the magnetic tape file system returns the RMS attributes to your process. For example, when you specify the DIRECTORY/FULL command for a volume, file, or list of files, the magnetic tape file system selects the file identifiers from the header labels, returns the file attributes to your process, and positions the magnetic tape after the header labels of the last file accessed.

A magnetic tape file opened for read access is closed either implicitly or explicitly. The file is closed implicitly when the driver encounters a tape mark while a file is being read. The magnetic tape file system then reads the trailer labels, closes the file, and positions the magnetic tape at the next file.

The file is closed explicitly when you deaccess the file before all the data in the file is read. The magnetic tape file system then closes the file without reading the trailer labels, and the magnetic tape remains at the current position.

4.3.2.3 Writing to Files on Magnetic Tape Volumes

When you use DCL to access an existing file for a write operation, one of the following operations is actually being performed: append or update. The difference between append and update write operations can be explained in the following way:

- Append Access

  When a file is accessed for an append operation, the magnetic tape is positioned at the EOF before the tape mark that precedes the trailer labels. After the file is appended and closed, all files beyond the appended file are lost. When the positioning is complete, the processing is handled as if the file had been created as described in the section on file creation.

- Update Access

  When a file is accessed for an update operation, the magnetic tape is positioned at the BOF section after the header labels. After the file is written to and closed, all files beyond the updated file are lost. The processing is handled as if the file had been created.

Note that you can update or append magnetic tape files only when the header label contains a value of zero for the buffer offset length. For more information on how to update and append magnetic tape files, see Chapter 5.

In addition to updating and appending files on a magnetic tape volume, you can access a volume for a write operation by using the CREATE command to write a new file to the magnetic tape volume. For example, you can enter the following command string:

```
$ CREATE MTA0:MYFILE
```

You can then write the contents of the file, without leaving the DCL command level, before closing the file.

If you do not specify the /OVERRIDE=EXPIRATION qualifier, the magnetic tape file system checks the expiration date field on the file before it allows you to write to that file. When more than one file is to be overwritten, the magnetic tape file system also checks the expiration date of the file immediately following the file to be overwritten. For example, before you append to a file, the magnetic tape file system checks the expiration dates of both the file being appended and the file immediately following. If the expiration date of either file has not been reached, the magnetic tape file system does not allow you to append the file.

When files are written to a magnetic tape volume, the magnetic tape file system performs access checks, writes labels, and, if necessary, switches volumes. If the new file will overwrite an existing file, the magnetic tape file system checks the expiration date and accessibility fields of the existing file. If overwriting is allowed, the magnetic tape file system overwrites the header label set of the existing file, creates the file section, writes the trailer labels, and writes two tape marks to denote the logical end-of-volume (EOV). All files following the newly created file are lost.

To close a magnetic tape file that was opened for WRITE access, the magnetic tape file system issues commands to the driver to write the labels, which are followed by a double tape mark that indicates the logical EOV.

4.4 Using Command Procedures to Access Foreign Volumes

The command procedures in this section allow you to create, read, or write magnetic tape data in a simple, user-defined, foreign format.

```
$!
$!              FOREIGN.COM
$!
$! This is the master command procedure. It sets up the user account and
$! mounts the volume with the /FOREIGN qualifier. If the user wants to
$! read a foreign volume, the FORREAD.COM command procedure is called.
$! If the user wants to write a foreign volume the FORWRITE.COM
$! command procedure is called.
$!
$       verify_off_on = F$VERIFY ( 0 )
$       ON CONTROL_Y THEN GOTO clean_up
$       ON WARNING THEN GOTO clean_up
$
$!
$! If VOLPRO privilege is not set but the user account has SETPRV, VOLPRO is
$! set to allow the user to mount a new unformatted volume.  If VOLPRO
$! privilege cannot be set, the user process is notified that the process
$! has insufficient privilege to write an unformatted volume. The user is also
$! asked whether to continue or exit the procedure.
$!
$       volpro_off_on = F$SETPRV ( "VOLPRO" )
$       IF F$PRIVILEGE ( "VOLPRO" ) THEN GOTO cont
$       WRITE SYS$OUTPUT "Insufficient Privilege to write an unformatted volume!"
$       INQUIRE/NOPUNC continue "Do you wish to continue (Y/N) ? "
$       IF .NOT. continue THEN GOTO clean_up
$ cont:
$
$!
$! Find out where the volume is mounted
$!
$ get_drive:
$       INQUIRE tape_drive "Tape drive"
$       IF tape_drive .EQS. "" THEN GOTO get_drive
$       tape_drive = tape_drive-":"+":"
$       IF .NOT. F$GETDVI (TAPE_DRIVE,"EXISTS")
$               THEN GOTO NOSUCHDEV
$       ASSIGN 'tape_drive' tape
$
```

```
$!
$! Try allocating and mounting the volume
$!
$       ALLOCATE tape:
$       MOUNT/NOASSIST/FOREIGN/OVERRIDE=(ACCESSIBILITY,EXPIRATION) tape:
$
$!
$! The user is asked whether a file will be read or written
$!
$
$ read_write:
$       INQUIRE/NOPUNC operation "Read or Write a file ? "
$       IF F$LOCATE ( operation, "READ" )  .EQ. 0 THEN GOTO read_op
$       IF F$LOCATE ( operation, "WRITE" ) .EQ. 0 THEN GOTO write_op
$       GOTO read_write
$
$ read_op:
$       @FORREAD
$       GOTO clean_up
$
$ write_op:
$       @FORWRITE
$
$ clean_up:
$
$!
$!  Reset the user account the way it was before the command procedure began
$!
$       SET NOON
$       DISMOUNT/NOUNLOAD tape:
$       DEALLOCATE tape:
$       DEASSIGN tape
$       volpro_off_on = F$SETPRV ( volpro_off_on )
$       verify_off_on = F$VERIFY ( verify_off_on )
$
$ EXIT
$
$ NOSUCHDEV:
$        WRITE SYS$OUTPUT "No Such Device"
$        GOTO get_drive
```

```
$!  FORWRITE.COM    Writes Data to a Foreign Volume
$!
$!  This command procedure writes data to a foreign volume.
$!  The data format is as follows:
$!       Each record is a block.
$!       Records are variable length.
$!
$!      The first four characters of a record are digits that are
$!      padded on the left with spaces. This sequence field starts at
$!      1 and increases by 1 with each record.
$!
$!      The fifth character of a record is a comma.
$!
$!      The sixth through ninth characters are digits that are padded on the
$!      left with spaces. This is a size field, which is the size of the data
$!      field.
$!
$!       The tenth character of a record is a vertical bar.
$!
$!       The rest of the record is data.
$!
$!       Records are padded to be at least 20 characters long.
```

```
$!
$
$        seq_num = 0    ! initialize the sequence number
$        spaces  = "           "   ! used to pad records less then 20 chars
$
$        OPEN/WRITE tape_file tape: ! open the output file
$ next_line:
$
$        seq_num = seq_num + 1  ! increment the sequence number
$
$!  Prompt the user for the data
$!       close the file and exit if ^Z is entered
$!
$        READ/PROMPT="Record # 'seq_num' : "/END_OF_FILE=end_of_input -
             SYS$COMMAND user_data
$
$!  Find the size of the record and the number of pad characters needed
$!  Note That a negative number of pad characters does not return any characters
$!
$        data_size = F$LENGTH ( user_data )
$        pad_chars = 10 - data_size
$        IF pad_chars .LT. 0 THEN pad_chars = 0
$
$!  Construct the output record using FAO
$!
$        out_rec = F$FAO ( "!4UL,!4UL,!AS", seq_num, data_size, user_data ) + --
           F$EXTRACT ( 1, pad_chars, spaces )
$
$        WRITE tape_file out_rec ! write the formatted output record
$        GOTO next_line
$
$ end_of_input:
$        CLOSE tape_file


$!             FORREAD.COM    Reads Data from a Foreign Volume
$!
$!  This command procedure is called when data will be read.  The procedure
$!  reads data on a foreign volume.  The data format is defined in
$!  FORWRITE.COM command procedure.
$!
$
$        seq_num = 0    ! initialize the sequence number
$
$        OPEN/READ tape_file tape: ! open the output file
$ next_line:
$
$         seq_num = seq_num + 1  ! increment the sequence number
$
$         READ/END_OF_FILE=end_of_input tape_file in_rec
$
$         record_num = F$EXTRACT ( 0, 4, in_rec ) -- " " -- " " -- " "
$         record_num = F$INTEGER ( record_num )
$         IF seq_num .NE. record_num THEN -
             WRITE SYS$OUTPUT "Error possible data lost, record sequence broken!"
$
```

```
$!
$!  Find the size of the data and extract the data
$!
$        data_size = F$EXTRACT ( 5, 4, in_rec ) -- " " -- " " -- " "
$        data_size = F$INTEGER ( data_size )
$        data_rec  = F$EXTRACT ( 10, data_size, in_rec )
$
$        WRITE SYS$OUTPUT data_rec ! write the data output record
$        GOTO next_line
$
$ end_of_input:
$        CLOSE tape_file
```

5 Transferring Information

The VMS operating system provides various facilities for transferring information contained on disk and magnetic tape media. In particular, the DCL command COPY and the VMS Convert and Exchange Utilities can be used to transfer disk and magnetic tape information.

5.1 Transferring Information Within and Across Operating Systems

In many cases, you will be able to transfer information without physically transporting media. You may, however, find it necessary to transfer files between systems that are not connected by a communications link. Under these circumstances, you must be able to physically move your files from one location to another. A convenient way to do this is to copy your files to a portable volume, such as a magnetic tape reel or disk pack, and then carry that volume to the location of the other system.

The VMS operating system supports the transfer of information contained on disks and magnetic tapes both within the VMS system and across other operating systems. It provides a number of facilities to assist you in both types of information transfer. The two most frequently used facilities for transferring information are the DCL command COPY and the VMS Exchange Utility (EXCHANGE).

In many cases, you will find the COPY command and the Exchange Utility sufficient for accomplishing information transfers. You may also find the VMS Convert and Analyze/RMS_File Utilities useful for transferring information, especially in foreign-volume or non-file-structured environments.

Under some circumstances, you will need to use the Backup Utility (BACKUP) to transfer files. When using magnetic tape, for example, BACKUP is the only means of transferring entire directory trees or files that are not sequentially structured.

Each of the information transfer facilities provided by the VMS system, except BACKUP, is described in the sections that follow. (See the *VMS Backup Utility Manual* for information on the use of BACKUP.)

5.2 Using the COPY Command to Transfer Information

One way of transferring information on disk or magnetic tape media is to use the DCL command COPY. The COPY command copies files from

- Disk to disk
- Disk to magnetic tape
- Magnetic tape to disk
- Magnetic tape to magnetic tape

The sections that follow describe how the COPY command is used with both disk and magnetic tape files.

5.2.1 Copying Files from Disk Volumes

This section describes how to use the COPY command to copy files from disk volumes to other disk volumes and from disk volumes to magnetic tape volumes.

The default format for files on disk volumes is called Files–11 Structure Level 2. You can also initialize disks in the Files–11 Structure Level 1 format, which is the format used by other DIGITAL operating systems including RSX–11M, RSX–11M–PLUS, RSX–11D, and IAS.

In the following example, the COPY commands copy the highest versions of all files in the current default directory and in the directories [PRIMER] and [COMMANDS] to the directory [PUBS]:

```
$ DEFINE P  DMA3:[PUBS]
$ COPY *.*  P:
$ COPY [PRIMER]*.*  P:
$ COPY [COMMANDS]*.*  P:
```

If you want to copy files from a disk directory — such as your default directory — on a public disk volume to a private Files–11 disk volume, you can also use the COPY command. Before copying these files, however, you must set up (allocate, initialize, and mount) a disk device.

In the following example, assume that the disk device DMA5 has been allocated to your process and that a disk volume has been initialized and mounted on that device. Also assume that you have a directory called PRIVATE already created on that volume. Copy the highest version of all the files in your default directory to the directory on that volume by entering the following command:

```
$ COPY *.*  DMA5:[PRIVATE]
```

You can also use the COPY command to copy files from a disk volume to a magnetic tape volume. The procedure for copying files from disk volumes to magnetic tape volumes is similar to those previously outlined for copying files across disk volumes. Note that magnetic tapes are sequential-access devices and do not have directories.

Again, you must set up (allocate, initialize, and mount) a magnetic tape device before copying disk files to the magnetic tape volume. In the following example, assume that MTA2 has been allocated to your process and that a magnetic tape volume has been initialized and mounted on that device. You can now use the COPY command to write files to the magnetic tape volume, as in the following example:

```
$ COPY *.*  MTA2:
```

In this case, the highest versions of all files in your default disk directory are copied to the magnetic tape volume on MTA2. The file names, file types, and version numbers of the output files default to the same file names, file types, and version numbers as the input files.

If you enter the COPY command with the /LOG qualifier, the system sends a message to the current SYS$OUTPUT device after each file has been copied. To verify that the files were successfully copied, use the DIRECTORY command. For example, the following command lists the file names and file types of all files on the magnetic tape volume:

```
$ DIRECTORY MTA2:
```

When you copy files from disk to ANSI-labeled volumes, the following items are not preserved:

- Directory specifications
- Individual file protection
- User identification code (UIC)
- Creation time (but the date is preserved)
- Revision and backup dates and times

5.2.2 Copying Files from Magnetic Tape Volumes

This section describes how to use the COPY command to copy files from magnetic tape.

The default format for files on magnetic tapes is the ANSI-labeled volume. The VMS system supports sequential, relative, and indexed files on disks, but only sequential files can be copied to ANSI-labeled volumes. The only valid record formats are variable-length (ANSI D) and fixed-length (ANSI F). In the example that follows, the ANSI file NEWINFO is copied from magnetic tape volume to the file TESTING.DAT:

```
$ COPY MTA1:NEWINFO    TESTING.DAT
```

Although the VMS system supports stream and variable with fixed-length control (VFC) records, it encodes these records in a variable-length format on ANSI-labeled volumes. Non-VMS systems do not distinguish stream records from VFC records; instead, they interpret both as variable-length records. Therefore, neither stream nor VFC records should be created on volumes that will be used for information interchange to a non-VMS system.

The full set of ASCII "a" characters are supported only for ANSI-labeled volumes, not for disk volumes. Therefore, when you copy files with ANSI file names from magnetic tape to disk, specify a standard VMS file name for the output file name specification.

If you do not specify a VMS file name on output, your process receives the following error message:

```
RMS-F-FNM, error in file name
```

This message indicates that the ANSI file name is not a valid file name specification.

The entire set of Files–11 file names is supported for magnetic tapes. You can copy a disk file with the following file name to a magnetic tape volume, without having to modify the file name:

THIS_IS$A_VAXVMSLONG_FILE.LONG_EXT

You can also use the Exchange Utility to perform file transfers and format conversions for DOS–11 magnetic tape volumes, Files–11 volumes, and RT–11 block-addressable volumes. Refer to Section 5.4 of this manual or to the *VMS Exchange Utility Manual* for more information.

5.2.2.1 Continuing the Copy Command at End-of-Tape

When you are copying to or from a magnetic tape and that tape reaches the end, the system suspends processing and sends a request to you to mount the next magnetic tape in the volume set. An Operator Communication Facility (OPCOM) message similar to the following may be displayed at your terminal:

```
%%%%%%%%%%% OPCOM, 14-JUN-1985 15:23:31.78 %%%%%%%%%%%
request 3, from user PLAW
MOUNT new relative volume 2 (DWOQT2) on MTA1:
```

Note: **Normally you do not see this message (messages may be sent only to the operator's terminal that has been enabled for tape messages), and you may not realize that another tape is needed to complete the read or write operation.**

See the *Guide to Maintaining a VMS System* for more information on OPCOM messages.

If automatic volume switching is disabled or if the magnetic tape file system cannot mount a given volume, you may need to mount a continuation volume in a volume set. See Section 3.4.4.2 for information on mounting a continuation volume. After loading the continuation volume on the drive specified in the mount request, mount the volume by entering the REPLY command with one of the following three qualifiers:

- /TO=request_id [volume identifier] — Used for both read and write copy operations. During a write operation, use the /TO qualifier if you want the volume identifier specified in the mount request to be written on the continuation volume.

  For example, to respond to the mount request 3, mount volume DW0QT2 on drive MTA1 and enter one of the following commands:

  ```
  $ REPLY/TO=3
  $ REPLY/TO=3 "DWOQT2"
  ```

  The first REPLY command does not specify a volume identifier; the second does.

- /INITIALIZE_TAPE — Used for write operations if the volume identifier on the continuation volume does not match the one specified in the mount request. The file system reinitializes the tape and mounts the volume with the new volume identifier. The magnetic tape file system then performs access checks and initializes the volume as if the INITIALIZE command had been specified. Any data on the tape prior to specifying the /INITIALIZE_TAPE command is lost. Either of the following REPLY commands is valid:

  ```
  $ REPLY/INITIALIZE_TAPE=3
  $ REPLY/INITIALIZE_TAPE=3 "DWOQT2"
  ```

  The first command does not specify a volume identifier; the second does.

- /BLANK_TAPE — Used to write to an unformatted volume. This qualifier initializes the volume and requires the VOLPRO and OPER privileges to avoid a runaway tape or timeout condition (see Chapter 3). Either of the following REPLY commands is valid:

  ```
  $ REPLY/BLANK_TAPE=3
  $ REPLY/BLANK_TAPE=3 "DWOQT2"
  ```

The first command does not specify a volume identifier; the second does. Specifying the volume identifier in either the MOUNT command or the REPLY/TO command is essential during write operations because it ensures that the correct volume is mounted on the drive and links the continuation volume to the volume set.

You can omit the volume identifier with the REPLY/TO command under two circumstances: (1) when reading from tape, the volume identifier is optional; (2) during a write operation, you must omit the volume identifier to preserve the accessibility code on a volume. If you initialize and mount a volume set in which each volume has a unique accessibility character that you want to maintain, avoid using the volume identifier since it causes the accessibility character of the first volume in the set to overwrite the accessibility code on the continuation volume. For example, to preserve the accessibility character, enter the following command, where 3 is the request identification number:

```
$ REPLY/TO=3
```

Once it receives the REPLY command, the magnetic tape file system performs checks on the continuation volume to ensure that it is the correct volume. As long as it is the correct volume with proper access codes, the system mounts the volume and reissues pending read or write requests to the continuation volume. If the volume fails any of these access checks, the volume is not mounted (or initialized and mounted, in the case of a blank tape).

The following examples illustrate ways of copying files to and from magnetic tape volumes.

Examples

1

```
$ COPY/LOG MTA1:"%&*?!SKI! """ SEASON.DAT
%COPY-S-COPIED, MTA1:[]"%&*?!SKI! """.;1
copied to WRKD:[MANUAL]SEASON.DAT;1 (120 records)
```

Since the /LOG qualifier was specified, the system returns a message that confirms the file was copied as specified and informs you how many records were copied. The ANSI file %&*?!SKI!#" (# means space) is copied to the file SEASON.DAT on the default disk and directory WRKD:[MANUAL]. The file could not have been copied to disk unless the new file name was specified. The VMS software provided defaults for segments of the file specification that were not specified.

2

```
$ COPY/LOG FORTAP.DAT MTA1:"%&*?!SKI! "" "
%COPY-S-COPIED, WRKD:[MANUAL]FORTAP.DAT;1
copied to MTA1:[]"%&*?!SKI! """.;0 (120 records)
```

In this command an ANSI file name was specified as the output file specification. Note that the trailing space in the file name %&*?!SKI!##"# (where # means space) is truncated.

3

```
$ COPY/LOG VAXVMS_LONG$FILE_NAME.LONG_EXT MTA1:
%COPY-S-COPIED, WRKD:[MANUAL]VAXVMS_LONG$FILE_NAME_EXT;1
copied to MTA1:VAXVMS_LONG$FILE_NAME.LONG_EXT;1 (80 records)
```

In this example, a VMS long file name with a long extension is copied to the volume MTA1 with the same file name and type that had been on the disk volume.

4
```
$ COPY %%.JOU;* MTA1:*.*
%COPY-S-COPIED, WRKD:[MANUAL]C6.JOU;1 copied to MTA1:[]C6.JOU;1 (4 records)
```

In this example, all files with a two-character file name and a file type of JOU are copied to the volume MTA1 with the same file name and type as they had on the disk volume. Version numbers are preserved.

5
```
$ COPY MTAO:*.* *
%COPY-S-COPIED, MTAO:[]TASTETEST.DAT;1
copied to WRKD:[FOOD]TASTETEST.DAT;1 (249 records)
%COPY-S-COPIED, MTAO:[]ALLAT;1 copied to WRKD:[FOOD]ALALL;1 (48 rcords)
%COPY-S-NEWFILES, 2 files created
```

In this example, neither file on the tape volume had an ANSI file name. Therefore, both files were copied to the disk volume.

6
```
$ COPY MTA1:*.* [EX]
%COPY-S-COPIED, MTA1:[].DAT;1 copied to WRKD:[EX]TEST.DAT21 records)
%COPY-E-OPENOUT, error opening WRKD:[EX]"%&*()!SKI! """.;1 as output
-RMS-F-FNM, error in file name
%COPY-W-NOTCOPIED, MTA1:[]"%&*()!SKI! """.;1 not copied
%COPY-E-OPENOUT, error opening WRKD:[EX]"SANFRAN%%%""".;1 as output
-RMS-F-FNM, error in file name
%COPY-W-NOTCOPIED, MTA1:[]"SANFRAN%%%""".;1 not copied
%COPY-S-COPIED, MTA1:[]VAXVMS_LONG$FILE_NAME.LONG_EXT;1
copied to WRKD$:[EX]VAXVMS_LONG$FILE_NAME.LONG_EXT;1 (80 records)
%COPY-S-COPIED, MTA1:[]C6.JOU;1 copied to WRKD:[EX]C6.JOU;1 (4 records)
%COPY-S-NEWFILES, 2 files created
```

The COPY command string specifies that all files on the volume mounted on drive MTA1 should be copied to the current default disk and directory WRKD:[EX]. However, files with ANSI file names are not copied; VMS returns an error message to the process.

The following sections provide guidelines for interchanging volumes.

5.2.3 Copying Files to and from Non-File-Structured Volumes

The VMS operating system supports the transfer of files between file-structured and non-file-structured volumes. You can use the COPY command to transfer files in the following ways:

- From file-structured volumes to non-file-structured volumes
- From non-file-structured volumes to file-structured volumes

The next two sections contain examples of each type of transfer.

5.2.3.1 Copying Files to a Non-File-Structured Volume

This section contains a procedure for copying files from a file-structured volume to a non-file-structured, or foreign volume.

The following procedure copies three files from an ANSI-labeled volume to a foreign volume.

1 Mount the volumes involved in the copy operation as follows:

```
$ MOUNT MTA1: FRESKI
%MOUNT-I-MOUNTED, FRESKI mounted on _MTA1:
$ MOUNT/FOREIGN MTAO:
%MOUNT-I-MOUNTED, mounted on _MTAO:
```

2 Enter the COPY command string, including the full device and file name specifications in the input specification, but specify the device name only in the output specification. File names are specified on input because the volume from which files will be copied is ANSI-labeled and file-structured. However, only the device name is specified on output because the volume to which the files will be copied is foreign and is not file-structured.

```
$ COPY/LOG MTA1:PROG1.EXE, PROG2.EXE, PROG3.EXE MTA0:
%COPY-S-COPIED, MTA1:[]PROG1.EXE;1 copied to MTA0: (92 records)
%COPY-S-COPIED, MTA1:[]PROG2.EXE;6 copied to MTA0: (70 records)
%COPY-S-COPIED, MTA1:[]PROG3.EXE;2 copied to MTA0: (77 records)
%COPY-S-NEWFILES, 3 files created
```

Because no version numbers were specified for the files copied from the ANSI-labeled magnetic tape volume, the first version of each specified file the MTAACP finds on the ANSI-labeled volume will be the version of the file that is copied to the foreign volume.

When you use the /LOG qualifier with the COPY command, the system informs you which files are copied and tallies the number of records copied and the number of files created.

When you copy files as shown in this example, the structure of the resulting magnetic tape is referred to as unblocked, non-file-structured. Each file record becomes a physical record on the magnetic tape. Each file boundary is indicated on the magnetic tape by a single tape mark. Two consecutive tape marks indicate end of volume. No label, file name, or attribute information is present.

5.2.3.2 Copying Files from a Non-File-Structured Volume

This section contains a procedure for copying data segments from a non-file-structured, or foreign, volume to a file-structured ANSI-labeled volume. Note that you must specify a COPY command for each segment of data that you copy from a foreign, or non-file-structured, volume to a file-structured volume.

1 Mount the volumes as follows:

```
$ MOUNT/NOLABEL MTA1: " " COLD:
%MOUNT-I-MOUNTED, mounted on _MTA1:

$ MOUNT MTA0: FEVER SEASON:
%MOUNT-I-MOUNTED, FEVER mounted on _MTA0:
```

2 Enter a COPY command for each segment on the foreign volume to be copied to files on the file-structured volume. Because the volume to which data segments are copied is file-structured, you must specify a file name in the output file specification.

```
$ COPY/LOG COLD: SEASON:FILE1.DAT
%COPY-S-COPIED, MTA1: copied to SEASON:FILE1.DAT;1 (92 records)
$ COPY/LOG COLD: SEASON:FILE2.DAT
%COPY-S-COPIED, MTA1: copied to SEASON:FILE2.DAT;1 (62 records)
$ COPY/LOG COLD: SEASON:FILE3.DAT
%COPY-S-COPIED, MTA1: copied to SEASON:FILE3.DAT;1 (23 records)
$ COPY/LOG COLD: SEASON:FILE4.DAT
%COPY-S-COPIED, MTA1: copied to SEASON:FILE4.DAT;1 (48 records)
$ COPY/LOG COLD: SEASON:FILE5.DAT
%COPY-S-COPIED, MTA1: copied to SEASON:FILE5.DAT;1 (37 records)
$ COPY/LOG COLD: SEASON:FILE6.DAT
%COPY-S-COPIED, MTA1: copied to SEASON:FILE6.DAT;1 (10 records)
```

Since the /LOG qualifier was used with the COPY command, VMS informs you that the files you specified were copied to the appropriate device and that tallies the number of records transferred.

5.3 Using the Convert Utility to Transfer Information

This section describes how to use the VMS Convert Utility (CONVERT) to transfer information to non-file-structured, or foreign, volume. It also describes how the VMS Analyze/RMS_File Utility (ANALYZE/RMS_FILE), which is useful in determining record size, is used in conjunction with CONVERT.

There may be cases where you will be unable to transfer information to a non-file-structured, or foreign, volume by using the COPY command. For example, files containing records smaller than 14 bytes cannot be copied successfully to a foreign volume with the COPY command. However, they can be transferred by using CONVERT.

CONVERT allows you to pad and extend the record size to make it suitable for a foreign volume. The following procedure illustrates how to use the Convert Utility:

1 Mount both volumes. Specify the /RECORDSIZE and /BLOCKSIZE qualifiers in addition to either the /NOLABEL or /FOREIGN qualifier when you mount the foreign volume.

```
$ MOUNT/FOREIGN/RECORDSIZE=80/BLOCKSIZE=2400 MTA1:
%MOUNT-I-MOUNTED, mounted on _MTA1:

$ MOUNT MTA0: LATER TODAY
%MOUNT-I-MOUNTED, LATER mounted on _MTA0:
```

The VMS operating system informs you that the foreign volume is mounted on drive MTA1 and that the target volume is mounted on MTA0. The /RECORDSIZE and /BLOCKSIZE qualifiers specify the tape is to be written as 80-byte records blocked in 2400-byte blocks.

2 Check the size of the largest record within a file. To do so, use the Analyze/RMS_File Utility as shown in the following example:

```
$ ANALYZE/RMS_FILE/STATISTICS TODAY:TICKET.DAT

RMS File Statistics                  14-JUN-1988 21:33:20.07
TODAY:[]TICKET.DAT;3
```

```
FILE HEADER
   File Spec: TODAY:[]TICKET.DAT;3
   File ID: (1,1,1)
   Owner UIC: [012,141]
   Protection: System:RWED, Owner:RWED, Group:WE, World:WE
   Creation Date:    5-APR-1988 14:20:04.06
   Revision Date:   17-NOV-1988 09:10:05.19, Number: 3
   Expiration Date: none specified
   Backup Date:     none posted
   Contiguity Options:  none
   Performance Options: none
   Reliability Options: none
   Journaling Enabled:  none

RMS FILE ATTRIBUTES

   File Organization: sequential
   Record Format: variable
   Record Attributes:   carriage-return
   Maximum Record Size: 255
   Longest Record: 77
   Blocks Allocated: 0, Default Extend Size: 0
   End-of-File VBN: 1, Offset: %X'0000'
   File Monitoring: disabled
   Global Buffer Count: 0

The analysis uncovered NO errors.

ANALYZE/RMS_FILE/STATISTICS TODAY:TICKET.DAT
```

3 Extend the size of records smaller than 14 bytes to that specified in the MOUNT command and ensure that the record size you specify is sufficient. To do so, enter the following commands:

```
$ CONVERT/PAD/FDL=SYS$INPUT
$_Input: TODAY:TICKET.DAT
$_Output: MTA1:
RECORD
SIZE 80
FORMAT FIX 15   EXIT
$
```

Entering the CONVERT command string, along with the input and output parameters, pads and extends the size of records when copying them from the ANSI-labeled volume mounted on MTA0 to the foreign volume mounted on MTA1. The CONVERT command and qualifiers can pad short records to correspond to the record size specified by the MOUNT command in the first step. After you enter the command line, CONVERT prompts you for input and output parameters. At the *$_Input:* prompt, supply the specification of the file containing the records that were too short. At the *$_Output:* prompt, specify only the physical or logical device name of the foreign volume. Pressing CTRL/Z terminates the file description and causes CONVERT to run. To verify that the file was copied to the foreign volume, you can rewind the foreign volume and type the volume as shown in the steps below.

4 To rewind a foreign volume, enter the following SET MAGTAPE /REWIND command string:

```
$ SET MAGTAPE/REWIND MTA1:
$
```

5 To verify the contents of a file, enter the TYPE command as follows:

```
$ TYPE MTA1:
This is a test file.
Some records contain less than 14 bytes.
This file was copied to a foreign volume with the
RMS Convert Utility.
Any record less than 80 bytes was extended and padded to
80 bytes.
```

The TYPE command types the first data segment on the volume. If you copy multiple files to a foreign volume with the RMS Convert Utility, each file is copied to the volume in a separate data segment. Therefore, you must enter a TYPE command for each data segment copied to the volume until you reach the segment that you want to verify.

This example produces what is referred to as a blocked non-file-structured volume. Each record of the input file is placed into an 80-byte area of a 2400-byte tape block. Tape marks are used to delimit files, as in the previous example.

5.4 Using the Exchange Utility to Transfer Information

The VMS system provides a utility called Exchange, which enables you to read to or write from disk and magnetic tape media being exchanged with DIGITAL-supported, non-VMS operating systems. With the VMS Exchange Utility (EXCHANGE), you can manipulate mass-storage volumes written in formats other than those normally recognized by VMS. You can also use EXCHANGE to manipulate Files–11 files that are images of foreign volumes.

You can use EXCHANGE to transfer files between foreign volumes and VMS file-structured volumes. You can also use EXCHANGE to perform volume-specific initialization and manipulation functions on the foreign volumes. The Exchange Utility enables you to convert the format of the files, as appropriate, when transferring files between volumes with different structures.

The Exchange Utility can perform file transfers and format conversions for the following:

- DOS–11 magnetic tape volumes
- Files–11 volumes
- RT–11 block-addressable volumes

In addition to transferring files, the Exchange Utility allows you to mount and dismount foreign volumes, initialize foreign volumes, list directories of volumes, delete files from block-addressable volumes, and locate bad blocks on the volume. For further information on how to use EXCHANGE to accomplish these tasks, see the *VMS Exchange Utility Manual.*

5.4.1 Invoking and Terminating the Exchange Utility

To invoke EXCHANGE, enter the following command in response to the DCL prompt:

```
$ EXCHANGE
EXCHANGE>
```

When you receive the EXCHANGE prompt (EXCHANGE>), you are in the Exchange Utility. Once you are in this utility, you can enter any of the command strings supported for EXCHANGE. For example:

```
EXCHANGE> DIRECTORY MFA0:/VOLUME=DOS11/FULL
```

This command lists all the files on the DOS–11 magnetic tape mounted on MFA0. In this case, the magnetic tape is rewound before the files are listed.

The following example illustrates the use of the MOUNT command within the Exchange Utility:

```
EXCHANGE> MOUNT DMA1:
%EXCHANGE-I-WRITELOCK, volume is write-locked
%EXCHANGE-I-MOUNTED, volume DMA1: mounted
```

This command mounts the foreign volume that is loaded in the RK07 device DMA1, making the volume available for subsequent commands. In this case, EXCHANGE recognizes that the volume itself is write-locked, and displays the message.

See the *VMS Exchange Utility Manual* for a complete list of all the commands, qualifiers, and parameters that are supported for EXCHANGE.

To exit from EXCHANGE and return to DCL level, use the EXCHANGE command EXIT or CTRL/Z.

5.4.2 Using EXCHANGE at DCL Command Level

In addition to using EXCHANGE interactively as a utility, you can use EXCHANGE as a DCL command. To do this, append a command string to the DCL command EXCHANGE as follows:

```
$ EXCHANGE DIRECTORY DMA1:/VOLUME_FORMAT=RT11
```

This DCL command lists the directory of the RT–11 volume mounted (/FOREIGN) on DMA1. At DCL level, EXCHANGE executes the single command and returns to the DCL prompt.

At DCL level, you can process only one command at a time. Some tasks for which EXCHANGE was designed, however, require more than one EXCHANGE command. For example, the use of virtual devices requires multiple EXCHANGE commands.

Tasks requiring more than one EXCHANGE command cannot be performed at DCL level. You can, however, design a command procedure to execute more than one EXCHANGE command for a particular task. In that case, you could execute the command procedure without leaving the DCL command level. For further details on how to construct command procedures using EXCHANGE, see Section 5.5.

5.5 Using Command Procedures to Transfer Information

You can design command procedures to facilitate routine transferring of information on disk and magnetic tape media.

Two examples of command procedures are included in this section. The first example is designed to transfer information by way of the COPY facility. The second example employs the Exchange Utility to transfer the information.

5.5.1 Using a Command Procedure to Copy Files

The sample command procedure in this section can be used for a variety of purposes. This procedure can copy from disk to similar disk, from disk to dissimilar disk, and from magnetic tape to magnetic tape. Note that the command procedure mounts the volume as non-file-structured.

If you are using magnetic tape media, this command procedure will not work for multi-reel magnetic tape volume sets. Magnetic tape to magnetic tape transfer is supported for single volumes only. The magnetic tape to magnetic tape copy is accomplished using the asterisk (*) wildcard character.

The sample command procedure is as follows:

```
$ vol_priv = f$setprv("volpro")
$ Type sys$input
$ on control_y then goto cleanup
$ on warning then goto cleanup

$ !
$ get_source:
$    write sys$output "Enter drive holding the write-locked master."
$    inquire source "[CS1$DRBO:, DBxx:, DDxx:, DJxx:, DLxx:, DMxx:, -
$    DQxx:, DRxx:, DUxx:, DYxx:, MFxx:, MTxx:, MUxx:,]"
$    if source .eqs. "" then goto get_source
$    source = source -":"+":"
$ !
$ get_target:
$    inquire target "Enter drive holding the write-enabled target."
$    if target .eqs. "" then goto get_target
$    target = target -":"+":"
$    kit_type := 'f$extract(0,1,source)
$    if f$locate("CS",source) .eq. f$length(source) .and. -
     f$locate("CS",target) .eq. f$length(target) then goto prepare
$
$ ! *Note* -- This section checks to see if the console is configured
$ !          into the system.
$ !          If it does not exist, SYSGEN will be run to configure it in
$ !
$ connect_console:
$    ! (NOTE* For 750,730 media copying CONSOLE must be configured)
$    if f$getdvi("CS:1","exists") then goto prepare
$    set noon
$    proc_priv = f$setprv("cmkrnl,cmexec")
$    run sys$system:sysgen
     connect console
$    proc_priv = f$setprv(proc_priv)
$    set on
$ !
$ prepare:
$    allocate ’source’
$    inquire start "Ready source volume ’’source’ for mounting and
$    press return"
$ !
$ begin:
$    allocate ’target’
$    mount/foreign ’source’
$    inquire start "Ready target volume ’’target’ for initialization and
$    press return"
$    volume_lbl = f$getdvi(source,"VOLNAM")
$    if (volume_lbl .eqs. "") then volume_lbl := "console"
$    init ’target’ ’volume_lbl’
$    If (kit_type .eqs. "M") then goto tape_copy
$    mount/foreign ’target’
$    backup/physical/verify ’source’ ’target’
$    goto copy_done
$ !
$ tape_copy:
$    mount/over:id ’target’
$    copy ’source’*.* ’target’*/log
$ !
$ copy_done:
$    dismount ’target’
$    deallocate ’target’
$    write sys$output "Copy is complete"
$    dismount/nounload ’source’
$    inquire answer "Do you want to make another copy of the Source volume?"
$    if .not. answer then goto new_source
$    write sys$output "Please place volume in the target device ’’target’." "
$    goto begin
$ !
$ new_source:
$    deallocate ’source’
$    inquire answer "Do you want to make a copy using a new Source device?
$    if answer then goto get_source
$    goto finish
$ !
$ cleanup:
$    if  f$getdvi(target,"mnt") then  dismount ’target’
$    if  f$getdvi(source,"mnt") then  dismount ’source’
$ !
$ finish:
$    vol_priv = f$setprv(vol_priv)
$    if f$getdvi(target,"all") then deallocate ’target’
$    if f$getdvi(source,"all") then deallocate ’source’
$    exit
```

5.5.2 Using a Command Procedure to Exchange Information

The following command procedure is designed to exchange files between the console device and the current directory on disk. The files to be copied are assumed to be in standard format as determined by file type.

```
$ WRITE SYS$OUTPUT " Command file to copy files to/from the system"
$ WRITE SYS$OUTPUT " console storage medium and the current directory."
$ WRITE SYS$OUTPUT " "
$ INQUIRE MOUNT "Is system console storage medium mounted (Y/N)?"
$ IF MOUNT THEN GOTO MOUNTED
$ WRITE SYS$OUTPUT "Please place the system console medium in the console drive"
$ INQUIRE MOUNT "and type RET when ready"
$ RUN SYS$SYSTEM:SYSGEN
CONNECT CONSOLE
$ MOUNT/SYSTEM/FOREIGN CSA1: "VAX console"
$ !
$ MOUNTED:
$ INQUIRE DIR "Copy from console medium (Y/N)?"
$ IF DIR THEN GOTO FROMCON
$ INQUIRE SOURCE "Enter file name(s)"
$ IF SOURCE .EQS. "" THEN GOTO EXIT
$ EXCHANGE COPY /LOG 'SOURCE' CSA1:
$ GOTO EXIT
$ !
$ FROMCON:
$ INQUIRE SOURCE "Enter console file name"
$ IF SOURCE .EQS. "" THEN GOTO EXIT
$ EXCHANGE COPY /LOG CSA1:'SOURCE' *
$ EXIT:
$ DISMOUNT CSA1:
$ MOUNT/SYSTEM/FOREIGN/NOWRITE CSA1: "VAX console"
```

A VMS Disk Files and Volumes

A.1 Files–11 Disk Structure

The VMS system recognizes two disk file structures: Files–11 On–Disk Structure Level 1 and Files–11 On–Disk Structure Level 2. Files–11 On–Disk Structure Level 2 is the default disk structure of the VMS system, and Files–11 On–Disk Structure Level 1 is a structure used by DIGITAL's RSX–11M, RSX–11D, RSX–11M–PLUS, and IAS operating systems.

Nine files control the structure of a Files–11 On–Disk Structure Level 2 volume. Only five of these files are used for a Files–11 On–Disk Structure Level 1 volume. Table A–1 identifies all nine files, which are referred to as reserved files, and indicates to which Files–11 On–Disk Structure Level they pertain.

Table A–1 VMS Reserved Files

Reserved File	File Name	Structure Level 1	Structure Level 2
Index file	INDEXF.SYS;1	X	X
Storage bit map file	BITMAP.SYS;1	X	X
Bad block file	BADBLK.SYS;1	X	X
Master file directory	000000.DIR;1	X	X
Core image file	CORIMG.SYS;1	X	X
Volume set list file	VOLSET.SYS;1		X
Continuation file	CONTIN.SYS;1		X
Backup log file	BACKUP.SYS;1		X
Pending bad block	BADLOG.SYS;1		X

All the files listed in Table A–1 are listed in the master file directory (MFD), [000000].

A.1.1 Index File

Every Files–11 volume has an index file, which is created when the volume is initialized. This index file identifies the volume to the operating system as a Files–11 structure and contains the access data for all files on the volume. The index file, which is listed in the master file directory as INDEXF.SYS;1, contains the following information:

- Bootstrap block — The volume's bootstrap block is virtual block number 1 of the index file. If the volume is a system volume, this block contains a bootstrap program that loads the operating system into memory. If the volume is not a system volume, this block contains a program that displays the message that the volume is not the system device but a device that contains users' files only.

- Home block — The home block establishes the specific identity of the volume, providing such information as the volume name and protection, the maximum number of files allowed on the volume, and the volume ownership information. The home block is virtual block number 2 of the index file.
- Alternate home block — The alternate home block is a copy of the home block. It permits the volume to be used even if the primary home block is destroyed.
- Alternate index file header — The alternate index file header permits recovery of data on the volume if the primary index file header becomes damaged.
- Index file bit map — The index file bit map controls the allocation of file headers and thus the number of files on the volume. The bit map contains a bit for each file header allowed on the volume. If the value of a bit for a given file header is 0, a file can be created with this file header. If the value is 1, the file header is already in use.
- File headers — The largest part of the index file is made up of file headers. Each file on the volume has a file header, which describes such properties of the file as file ownership, creation date and time, file protection, and location of the data in the file. The file header contains all the information needed for gaining access to the file.

A.1.2 Storage Bit Map File

The storage bit map file controls the available space on a volume; this file is listed in the master file directory as BITMAP.SYS;1. It contains a storage control block, which consists of summary information intended to optimize the Files–11 space allocation, and the bit map itself, which lists the availability of individual blocks.

A.1.3 Bad Block File

The bad block file, which is listed in the master file directory as BADBLK.SYS;1, contains all the bad blocks on the volume. The system detects bad disk blocks dynamically and prevents their reuse once the files to which they are allocated have been deleted.

A.1.4 Master File Directory

The master file directory (MFD) itself is listed in the MFD as 000000.DIR;1. The MFD, which is the root of the volume's directory structure, lists the reserved files that control the volume structure and may list both users' files and users' file directories. Usually, however, the MFD is used to list the reserved files and users' file directories; users seldom enter files in the MFD, even on private volumes. In fact, on a private volume, it is most convenient for a user to create a directory that has the same name as the user's default directory on a system disk. For an explanation of users' file directories and file specifications, see the *VMS DCL Concepts Manual.*

When the VMS Backup Utility (BACKUP) creates sequential disk save sets, it stores the save set file in the MFD.

A.1.5 Core Image File

The core image file is listed in the MFD as CORIMG.SYS;1. It is not supported by the VMS operating system.

A.1.6 Volume Set List File

The volume set list file is listed in the MFD as VOLSET.SYS;1. This file is used only on relative volume 1 of a volume set. The file contains a list of the labels of all the volumes in the set and the name of the volume set.

A.1.7 Continuation File

The continuation file is listed in the MFD as CONTIN.SYS;1. This file is used as the extension file identifier when a file crosses from one volume to another volume of a loosely coupled volume set. This file is used for all but the first volume of a sequential disk save set.

A.1.8 Backup Log File

The backup log file is listed in the MFD as BACKUP.SYS;1. This file is reserved for future use.

A.1.9 Pending Bad Block Log File

The pending bad block log file is listed in the MFD as BADLOG.SYS;1. This file contains a list of suspected bad blocks on the volume that are not listed in the bad block file.

A.1.10 Files–11 On–Disk Structure Level 1 Versus Structure Level 2

For reasons of performance and reliability, Files–11 On–Disk Structure Level 2, a compatible superset of Structure Level 1, is the preferred disk structure on the VMS system. At volume initialization time (see the INITIALIZE command in the *VMS DCL Dictionary*), Structure Level 2 is the default. Structure Level 1 should be specified only for volumes that must be transportable to RSX–11M, RSX–11D, RSX–11M–PLUS, and IAS systems, as these systems support only that structure level. Additionally, you may be required to handle Structure Level 1 volumes transported to VMS from one of the above systems.

Where Structure Level 1 volumes are in use on the system, you should bear in mind the following limitations on them:

- Directories — No hierarchies of directories and subdirectories, and no ordering of directory entries (that is, the file names) in any way. RSX–11M, RSX–11D, RSX–11M–PLUS, and IAS systems do not support subdirectories and alphabetical directory entries.
- Disk quotas — Not supported.
- Multivolume files and volume sets — Not supported.
- Placement control — Not supported.

- Caches — No caching of file header blocks, file identification slots, or extent entries.
- System disk — Cannot be a Structure Level 1 volume.
- VAXcluster access — Local access only; cannot be shared across a cluster.
- Clustered allocation — Not supported.
- Backup home block — Not supported.
- Protection code E — Means extend for the RSX–11M operating system but is ignored by VMS.
- File versions — Limited to 32,767; version limits are not supported.
- Enhanced protection features (for example, Access Control Lists) — not supported.
- Long file names — Not supported.
- RMS journaling — Not supported.
- RMS execution statistics monitoring — Not supported.

Future enhancements to VMS software will be based primarily on Structure Level 2, so that further restrictions on Structure Level 1 volumes might be incurred.

B VMS ANSI-Labeled Magnetic Tape

This appendix briefly describes ANSI labels, data, and record formats supported by the VMS operating system. It also describes support for the various ANSI labels on VMS. Note, however, that the VMS software also supports the ISO standard. For a complete description of these labels, please refer to the ANSI X3.27–1978 or ISO 1001–1979 standard.

B.1 Logical Format of ANSI-Labeled Volumes

The format of VMS ANSI-labeled magnetic tape volumes is based on Level 3 of the ANSI standard for magnetic tape labels and file structure for information interchange. This standard specifies the format, content, and sequence of volume labels, file labels, and file structures. According to this standard, volumes are written and read on 9-track magnetic tape drives only. The contents of labels must conform to prescribed data and record formats. All labels must consist of ASCII "a" characters.

The VMS ANSI-labeled format allows you to write binary data in the file sections (see Figure B–1) of files. However, if you plan to use such files for information interchange across systems, make sure that the recipient system can read the binary data.

B.2 VMS Magnetic Tape Ancillary Control Process (MTAACP)

The VMS magnetic tape ancillary control process (MTAACP) is the internal software process of the operating system that interprets the logical format of VMS ANSI-labeled volumes. Transparent to your process, the MTAACP process reads, writes, and interprets VMS ANSI labels before passing this information to VMS Record Management Services (RMS) and Queue I/O ($QIO) system services. These services in turn read, write, and interpret the record format of the data written in the file section.

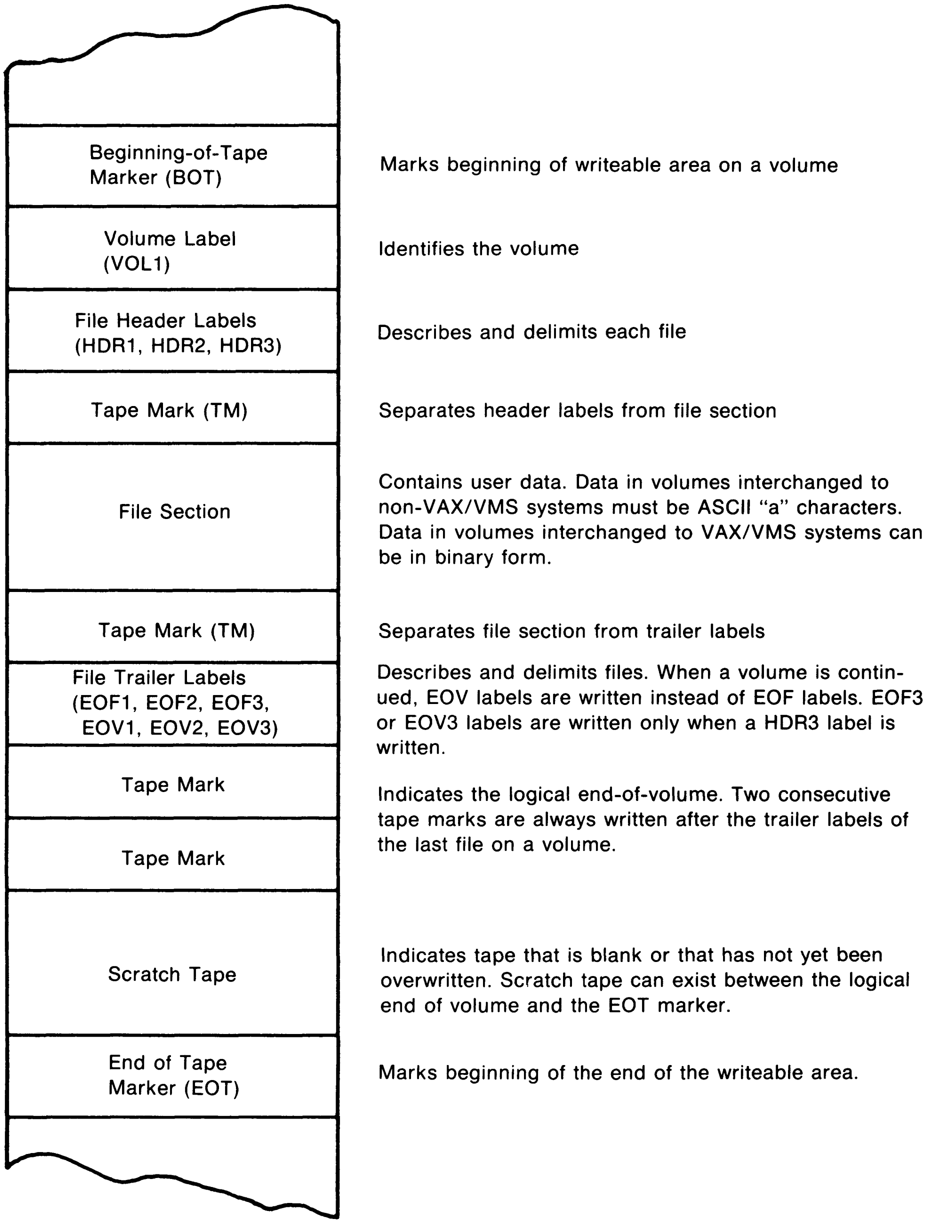
B.3 Basic Components of the VMS ANSI-Labeled Format

The format of VMS ANSI-labeled magnetic tape consists of four basic components:

- Beginning-of-tape (BOT) and end-of-tape (EOT) markers
- Tape marks
- File sections
- Volume, header, and trailer labels

Figure B–1 displays the arrangement and function of these components.

Figure B–1 Basic Layout of a VMS ANSI-Labeled Volume



Beginning-of-Tape Marker (BOT)	Marks beginning of writeable area on a volume
Volume Label (VOL1)	Identifies the volume
File Header Labels (HDR1, HDR2, HDR3)	Describes and delimits each file
Tape Mark (TM)	Separates header labels from file section
File Section	Contains user data. Data in volumes interchanged to non-VAX/VMS systems must be ASCII "a" characters. Data in volumes interchanged to VAX/VMS systems can be in binary form.
Tape Mark (TM)	Separates file section from trailer labels
File Trailer Labels (EOF1, EOF2, EOF3, EOV1, EOV2, EOV3)	Describes and delimits files. When a volume is continued, EOV labels are written instead of EOF labels. EOF3 or EOV3 labels are written only when a HDR3 label is written.
Tape Mark	Indicates the logical end-of-volume. Two consecutive tape marks are always written after the trailer labels of the last file on a volume.
Tape Mark	
Scratch Tape	Indicates tape that is blank or that has not yet been overwritten. Scratch tape can exist between the logical end of volume and the EOT marker.
End of Tape Marker (EOT)	Marks beginning of the end of the writeable area.

ZK-981-82

B.3.1 Beginning-of-Tape and End-of-Tape Markers

Every volume has beginning-of-tape (BOT) and end-of-tape (EOT) markers. These markers are pieces of photoreflective tape that delimit the writable area on a volume. ANSI standards require that a minimum of 14 feet to a maximum of 18 feet of magnetic tape precede the BOT marker; a minimum of 25 feet to a maximum of 30 feet of magnetic tape, of which 10 feet must be writable, must follow the EOT marker. The EOT marker indicates the start of the end of the writable area of the tape, rather than the physical end of the tape. Therefore, data and labels can be written after the EOT marker.

B.3.2 Tape Marks

Tape marks separate the file labels from the file sections, separate one file from another, and denote the logical end-of-volume. On the basis of the number and relative placement of tape marks written on a volume, the VMS system determines whether a tape mark delimits a label, a file, or a volume.

Tape marks are written both singly and in pairs. Single tape marks separate either a file section from the header and trailer labels or one file from another. When written after a set of header labels, a single tape mark signals the beginning of a file section. When written before a set of trailer labels, a single tape mark indicates the end of a file section. When written after a trailer label set, a single tape mark separates one file from another.

Double tape marks indicate that either an empty file section exists or the logical end-of-volume has been reached. The VMS system creates an empty file when a volume is initialized.

B.3.3 Labels

Labels identify, describe, and control access to volumes and their files. The VMS ANSI-labeled format supports volume, header, and trailer labels. The volume labels are the first labels written on a volume. They identify the volume and the volume owner and specify access protection. Header and trailer labels are sets of labels that identify, describe, and delimit files. Header labels precede files; trailer labels follow files.

Table B–1 lists the labels supported by the VMS operating system. All other ANSI labels are ignored by VMS on input.

Although each type of label uses a different format to organize its contents, all labels conforming to Version 3 of the ANSI standard must consist of ASCII "a" characters. Some labels contain reserved fields designed for future system use or future ANSI standardization. Reserved fields also must consist of ASCII "a" characters; however, VMS ignores these characters on input.

B.4 Volume and File Configurations

VMS ANSI-labeled volumes support four file/volume configurations:

- Single-file/single-volume
- Single-file/multivolume
- Multifile/single-volume
- Multifile/multivolume.

All these configurations conform to the following guidelines:

- The file sequence number field allows as many as 9999 file sections for one file. In effect, the file length is unlimited.
- Only one file section of a given file is written on a volume.

- When multiple sections exist for one file, each file section is written to a separate volume in the volume set. The file section numbers of each section are written consecutively in ascending order (section *n+1* is written immediately following section *n*); file sections of other files are not interspersed.

Each of the file/volume configurations is illustrated in the subsections that follow.

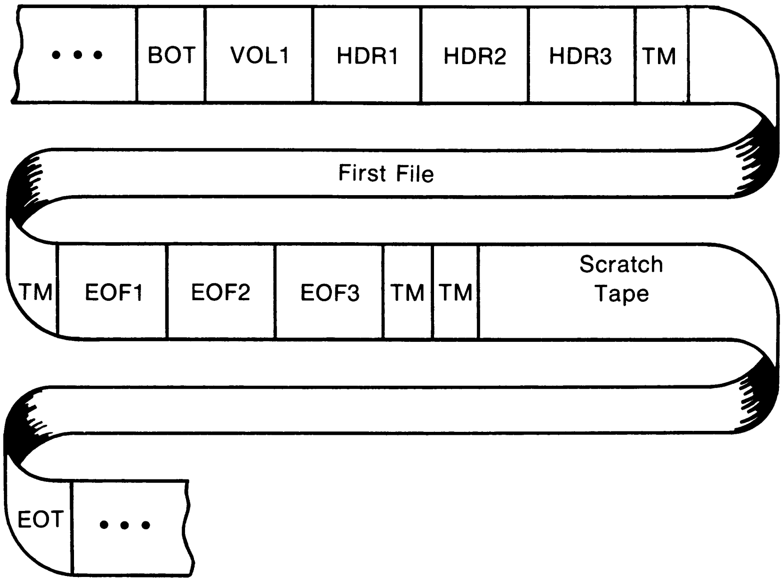
Table B–1 Labels and Components Supported by VMS

Symbol	Meaning
BOT	Beginning-of-tape marker
EOF1	First end-of-file label
EOF2	Second end-of-file label
EOF3	Third end-of-file label
EOF4	Fourth end-of-file label
EOT	End-of-tape marker label
EOV1	First end-of-volume label
EOV2	Second end-of-volume label
EOV3	Third end-of-volume label
EOV4	Fourth end-of-volume label
HDR1	First header label
HDR2	Second header label
HDR3	Third header label
HDR4	Fourth header label
VOL1	First volume label
VOL2	Second volume label
TM	Tape mark
TM TM	Double tape mark indicates an empty file section or the logical end-of-volume

B.4.1 Single-File/Single-Volume Configuration

A single-file/single-volume configuration consists of one file on one volume. The components of the ANSI-labeled format for this configuration are illustrated in Figure B–2.

Figure B–2 Single-File/Single-Volume Configuration



ZK-346-81

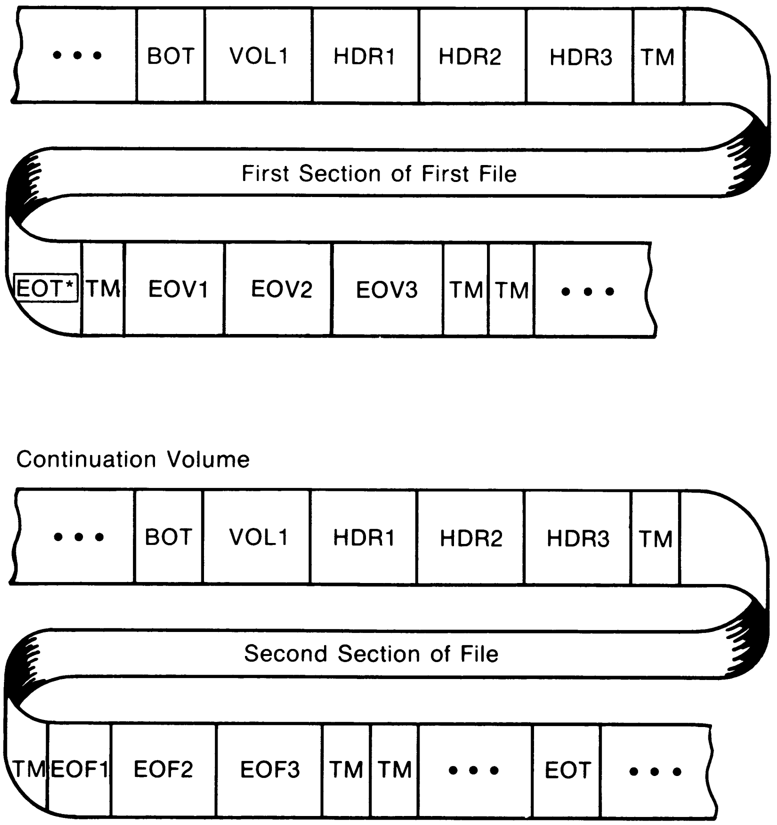
B.4.2 Single-File/Multivolume Configuration

A single-file/multivolume configuration consists of one file that spans two or more volumes in a volume set. Figure B–3 illustrates the components of the ANSI-labeled format for this configuration.

B.4.3 Multifile/Single-Volume Configuration

A multifile/single-volume configuration consists of two or more files on a single-volume. It is the most common file and volume configuration. Figure B–4 illustrates the components of the ANSI-labeled format for this configuration.

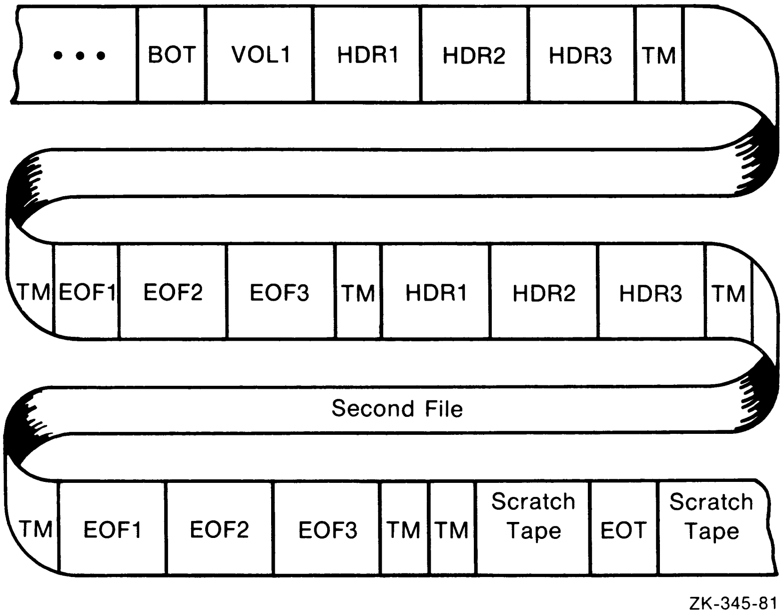
Figure B–3 Single-File/Multivolume Configuration



*When the driver encounters an EOT marker during a write operation, the MTAACP writes the appropriate trailer labels and performs a volume switch, if necessary.

ZK-347-81

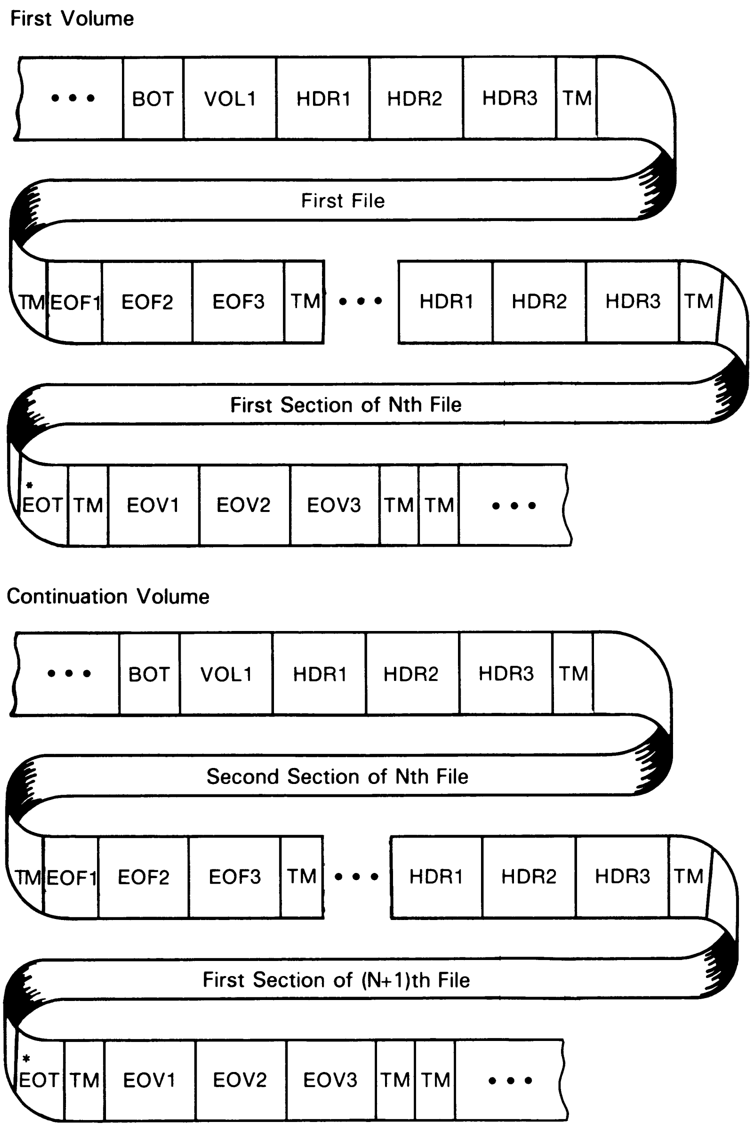
Figure B–4 Multifile/Single-Volume Configuration



B.4.4 Multifile/Multivolume Configuration

A multifile/multivolume configuration consists of two or more files that span two or more volumes in the same volume set. Figure B–5 illustrates the components of the ANSI-labeled format for this configuration.

Figure B–5 Multifile/Multivolume Configuration



*When the driver encounters an EOT marker during a write operation, the MTAACP writes the appropriate trailer labels and performs a volume switch, if necessary.

ZK-348-81

B.5 Volume Labels

The sections that follow describe the first volume (VOL1) and second volume (VOL2) labels.

B.5.1 VOL1 Label

The 80-character volume label (VOL1) is the first label written on a VMS ANSI-labeled volume. It defines the label type, name, and owner of the volume. Although there are many fields in a VOL1 label, this section describes only those fields that you can access or that can inhibit access to a volume and its files on VMS.

B.5.1.1 Volume Identifier Field

The volume identifier field is a 6-character field that contains the name of the volume. You specify the volume identifier in the command string when you initialize or mount a volume (see Chapter 3). The volume identifier consists of six ASCII "a" characters. Lowercase characters are not in the "a" set, but if you specify them, VMS will change them to uppercase. If you specify fewer than six characters, VMS pads the field by right-justifying the field with the ASCII space character.

B.5.1.2 Accessibility Field

The accessibility field is a one-character field that allows an installation to control access to a volume. See Section 2.1.3 for a description of accessibility support.

B.5.1.3 Implementation Identifier Field

The implementation identifier field contains the identifier of the implementation that creates the magnetic tape. This field controls how certain implementation-specific fields and volume labels are interpreted. The magnetic tape file system's implementation identifier is DECFILE11 A.

B.5.1.4 Owner Identifier Field

The owner identifier field is available to the user. This field does not affect the checking of a user's access to a volume, except as noted in Chapter 3.

B.5.2 VOL2 Label

In addition to the first volume (VOL1) label described above, VMS also provides a second volume (VOL2) label, the volume-owner field.

The volume-owner field contains the VMS protection information that has been written on the magnetic tape. A second volume label is written only if a VMS protection scheme had been specified on either the MOUNT or INITIALIZE command.

The volume-owner field also contains a value that incorporates the user identification code (UIC) with the VMS protection code specified for a volume. By default, VMS does not write a UIC to this field, thus allowing all users READ and WRITE access. Note, however, that EXECUTE and DELETE access are not applicable to magnetic tape volumes. Also note that, regardless of the protection code that you specify, both system users and the volume owner always have READ and WRITE access to a volume. The contents of the volume-owner field depends on the VMS protection code that you specify.

B.6 Header Labels

The VMS operating system supports four file-header labels: HDR1, HDR2, HDR3, and HDR4. The HDR3 and HDR4 labels are optional. The following sections describe and illustrate each file-header label.

B.6.1 HDR1 Label

Every file on a volume has a HDR1 label, which identifies and describes the file by supplying the VMS MTAACP with the following information:

- File identifier
- File-set identifier
- File section number
- File sequence number
- Generation and generation version numbers
- File creation and expiration dates
- Accessibility code
- Implementation identifier

B.6.1.1 File Identifier Field

The file identifier field contains the first 17 characters of the file name you specify. The remainder of the file name is written into the HDR4 label, provided that this label is allowed. If no HDR4 label is supported, a file name longer than 17 characters will be truncated. You may use either an ANSI file name or a VMS file specification of the following format:

```
filename.type;version
```

VMS file specifications are a subset of ANSI file names. However, ANSI file names are valid only for magnetic tape volumes; VMS file specifications are valid for disk and tape volumes. Both types of file specifiers are compatible with compatibility mode.

A VMS file specification consists of a file name, a file type, and an optional version number. Valid file names contain a maximum of 39 characters. Valid file types consist of a period followed by a maximum of 39 characters. The semicolon separates the version number from the file type.

Except for wildcard characters, only the characters A through Z, 0 through 9, and the special characters ampersand (&), hyphen (-), underscore (_) and dollar sign ($) are valid for VMS file names and types. The period and semicolon are the only other valid special characters, and they are always separators.

ANSI file names do not have a file type field. An ANSI file name consists of a 17-character name string, a period, a semicolon, and an optional version number. You can specify a name string consisting of a maximum of 17 ASCII "a" characters, but you must enclose the string in quotation marks (as in, for example, "file name"). When you specify fewer than 17 characters, the string is padded on the right with spaces to the 17-character maximum size. If you specify a file name that has trailing spaces, VMS truncates them when the file name is returned. In addition, the space-padded field prevents you from specifying a unique file name that consists of spaces.

Although you can specify longer file names (up to 79 characters), only the first 17 characters of the file name will be used in interchange.

The quotation mark character requires special treatment because it is both the file name delimiter and a valid ASCII "a" character that can itself be embedded in the name string. You must specify two quotation marks for each one that you want the VMS system to interpret. The additional quotation mark informs VMS that one of the quotation marks is part of the name string, rather than a delimiter.

Embedded spaces also are valid characters, but embedded tabs are not. Lowercase characters are not in the ASCII "a" character set, but if you specify them, VMS converts them to uppercase characters.

If you do not specify a file type or version number on input, VMS will supply a period (the default file type) and a semicolon (the default version number). However, the period and semicolon will not be written to this field on the tape.

Although the VMS operating system considers version numbers for ANSI and VMS file names to be part of the file name specification, the version number of a file is not written to the file identifier field but is mapped to the generation number and generation version-number fields as described in Section B.6.1.4.

Examples below display ANSI file names. The input is the format that you specify. The output shown displays the VMS format returned to your process and the format written to the label. The number sign (#) in the examples indicates a space character. In the last example, a VMS file name is enclosed in quotation marks, like an ANSI file name, on input. However, the VMS operating system returns the file name to the process as a VMS file name, rather than an ANSI file name. Therefore, when you enclose a valid VMS file name in quotation marks on input, VMS parses the file name as a VMS file name.

```
     Input

"AB2&D""FgHI*k4""#-M";2

"##########"

"""""""""""""""""""""""""""";

"DWDEVOP.DAT"

"VMS_LONG_FILENAME.LONG_FILETYPE"


   Output to User Process

"AB2&D""FGHI*K4""#-M";2

"".;

"""""""""""""""""""""""""""".;

DWDEVOP.DAT;

VMS_LONG_FILENAME.LONG_FILETYPE


   Output to HDR1 Label

AB2&D"FGHI*K4"#-M
```

```
#################

"""""""""""""####

DWDEVOP.DAT######

VMS_LONG_FILENAME
```

B.6.1.2 File-Set Identifier Field

The 6-character file-set identifier field denotes all files that belong to the same volume set. The file-set identifier for any file within a given volume set should always be the same as the file-set identifier of the first file on the first volume that you mount. For VMS, the file-set identifier is the same as the volume identifier of the first volume that you mount.

B.6.1.3 File Section Number and File Sequence Number Fields

The file section number is a 4-character field that specifies the number of the file section.

The file sequence number is a 4-character field that specifies the number of the file in a file set.

B.6.1.4 Generation Number and Generation Version-Number Fields

The generation number (a decimal number from 0001 to 9999) and generation version-number (a 2-digit decimal number) fields store the file version number specified on input and written by the system on output. The VMS operating system does not increment the version number of a file, even when the version of the specified file already exists on the volume. Therefore, if the file that you specify has the same file name and version number as an existing file, you will have at least two files with the same version number on the same volume set.

On input, VMS computes the version number by using this calculation:

```
version number =
[(generation number - 1) * 100] + generation version-number + 1
```

Version numbers larger than 32,767 are divided by 32,768; the integer remainder becomes the version number.

On output, the generation number is derived from the version number with this calculation:

```
generation number = [(version number - 1)/100] + 1
```

If there is a remainder after the version number is divided by 100, the remainder becomes the generation version number. It is not added to 1 to form the generation number.

B.6.1.5 Creation Date and Expiration Date Fields

The creation date field contains the date the file is created. The expiration date field contains the date the file expires. The system interprets the expiration date of the first file on a volume as the date that both the file and the volume expire. The creation and expiration dates are stored in the Julian format. This 6-character format (#YYDDD) consists of a space (#), the year, and the day. Only dates are relevant for these fields; time is always returned as 00:00:00:00.

By default, the current date is written to both the creation and expiration date fields when you create a file. Because the expiration date is the same as the creation date, the file expires on creation and you can overwrite it immediately. If the expiration date is a date that is later than the creation date and if the files you want to overwrite have not expired, you must specify the /OVERRIDE=EXPIRATION qualifier with the INITIALIZE or MOUNT command.

To write dates other than the VMS defaults in the date fields in this label, specify the creation date field (CDT) and the expiration date field (EDT) of the RMS date and time extended attribute block (XABDAT).

When you do not specify a creation date, VMS RMS defaults the current date to the creation date field. To specify a zero creation date, you must specify a year before 1900. If you specify a binary zero in the date field, the system will write the current date to the field.

For details on the XABDAT, see the *VMS Record Management Services Manual.*

B.6.1.6 Accessibility Field

The contents of this field are described in Section B.5.1.2.

B.6.1.7 Implementation Identifier Field

The implementation identifier field specifies, using ASCII "a" characters, an identification of the implementation that recorded the Volume Header Label Set.

B.6.2 HDR2 Label

The HDR2 label describes the record format, maximum record size, and maximum block size of a file.

B.6.2.1 Record Format Field

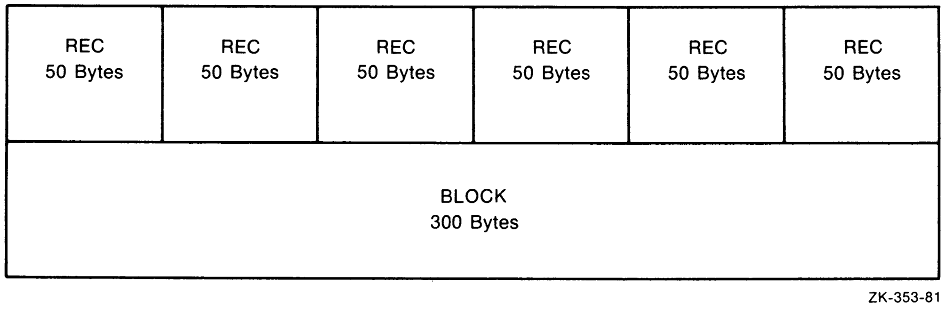
The record format field specifies the type of record format the file contains. The VMS operating system supports two record formats: fixed length (F) and variable length (D). When files contain record formats that the system does not support, it cannot interpret the formats and classifies them as undefined.

Fixed-length records are all the same length. No indication of the record length is required within the records because either the HDR2 label defines the record length or you specify the record length with the /RECORDSIZE qualifier. A fixed-length record can be a complete block, or several records can be grouped together in a block.

Fixed-length blocked records are padded to a multiple of 4 records. Variable-length records are padded to the block size. If a block is not filled, it will be padded with circumflex characters (^). The standard does not allow records containing only circumflexes; the system will interpret this as padding, not data.

Figure B–6 shows a block of fixed-length records. Each record has a fixed length of 50 bytes. All six records are contained in a 300-byte block. The records are blocked—that is, grouped together as one entity—to increase processing efficiency; when records are blocked, you can access many of them with one I/O request. The block size should be a multiple of the record size.

Figure B–6 Blocked Fixed-Length Records

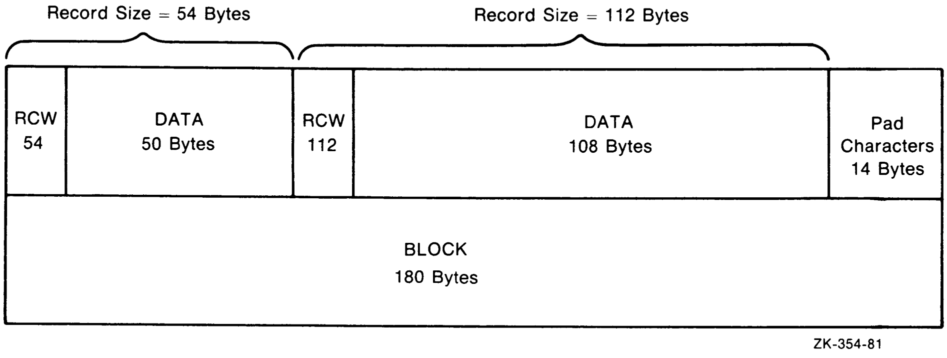


The size of a variable-length record is indicated by a record control word (RCW). The RCW consists of four bytes at the beginning of each record. A variable-length record can be a complete block, or several records can be grouped together in a block.

Two variable-length records are shown in Figure B–7. The first consists of 54 bytes, including the RCW. The second consists of 112 bytes, including the RCW. The records are contained in a 166-byte block.

Do not use system-dependent record formats on volumes used for information interchange. VMS system-dependent formats are stream and variable with fixed-length control (VFC).

Figure B–7 Variable-Length Records



B.6.2.2 Block Length Field

The block length field denotes the maximum size of the blocks. According to the ANSI standard, valid block sizes range from 18 to 2048 bytes. However, the VMS operating system allows you to specify a smaller or larger block size by using the /BLOCKSIZE qualifier with the MOUNT command. To specify the block size using VMS RMS, see the BLS field in the file access block (FAB) in the *VMS Record Management Services Manual.* When you specify a block size outside the ANSI standard range, the volume may not be processed correctly by other systems that support the ANSI standards.

B.6.2.3 Record Length Field

The record length field denotes either the size of fixed-length records or the maximum size of variable-length records in a file. Valid VMS record sizes vary, depending on the record format. The range for fixed-length records is 1 to 65,534 bytes; the range for variable-length records is 4 to 9,999 bytes, including the 4-byte RCW. Therefore, the maximum length of the data area of a variable-length record is 9,995 bytes. To comply with ANSI standards, the record size should not be larger than the maximum block size of 2,048 bytes, even though VMS allows larger record sizes (when the block size is larger).

For volumes containing files that do not have HDR2 labels, you must specify MOUNT/RECORDSIZE=n at mount time. The variable *n* denotes the record length in bytes. Files without HDR2 labels were created by a system that supports only level 1 or 2 of the ANSI standard for magnetic tape labels and file structure. Records should be fixed length because this is the only record format that either level supports. If you do not specify a record size, each block will be considered a single record.

B.6.2.4 Implementation-Dependent Field

The implementation-dependent field contains two 1-character subfields that describe how the VMS operating system interprets record format and form control.

The first subfield, character position 16, denotes whether the RMS attributes are in this label or the HDR3 label. If character position 16 contains a space, the RMS attributes are in the HDR3 label; if it contains any character other than a space, character position 16 is the first byte of the RMS attributes in the HDR2 label. The attributes appear in character positions 16 through 36 and 38 through 50.

For volumes created by any VMS version up to and including Version 2.0, the system-dependent fields of the HDR2 label contain the RMS file attributes in binary format. For volumes created on VMS versions later than Version 2.0 (Version 2.1 to the present), the magnetic tape file system stores the RMS attributes.

The second subfield, the form control field at character position 37, specifies the form control that defines the carriage control applied to records within a file. Possible values supported for VMS magnetic tape volumes are listed below.

A First byte of record contains FORTRAN control characters.

M The record contains all form control information.

space Line-feed/carriage-return combination is to be inserted between records when the records are written to a carriage-control device, such as a line printer or terminal. If form control is not specified when a file is created, this is the default.

B.6.2.5 Buffer-Offset Length Field

For implementations that support buffer offsets, the buffer-offset length field indicates the length of information that prefixes each data block. The magnetic tape file system supports the input of buffer offset, which means that the buffer-offset length obtained from the HDR2 label (when reading the file) is used to determine the actual start of the data block. The magnetic tape file system does not support the writing of a buffer offset.

Note that, if you open a file for append or update access and the buffer-offset length is nonzero, the open operation will not succeed.

B.6.3 HDR3 Label

The HDR3 label describes the RMS file attributes. For RMS operations, data in the HDR3 label supersedes data in the HDR2 label.

Although the HDR3 label usually exists for every file on a VMS ANSI-labeled volume, there are two situations when this label will not be written. The first is when an empty dummy file is created during volume initialization; no HDR3 label is written because the empty file does not require RMS attributes. The second is when you specify MOUNT/NOHDR3 at mount time. You should use the /NOHDR3 qualifier when you create files on volumes that will be interchanged to systems that do not process HDR3 labels correctly.

The RMS attributes describe the record format of a file. These attributes are converted from 32 bytes of binary values to 64 bytes of ASCII representations of their hexadecimal equivalents for storage in the HDR3 label.

B.6.4 HDR4 Label

The HDR4 label contains the remainder of a VMS file name that would not fit in the HDR1 file identifier field.

B.7 Trailer Labels

The VMS operating system supports two sets of trailer labels: end-of-file (EOF) and end-of-volume (EOV). A trailer label is written for each header label.

EOF and EOV labels are identical to their file header label counterparts except that:

- The label identifier field (characters 1-3) contains EOF or EOV.
- The block count field (characters 55-60) in the EOF1 and EOV1 labels contains the number of data blocks in the file section.

The particular label that VMS writes depends on whether a file extends beyond a volume. If a file terminates within the limits of a volume, EOF labels are written to delimit the file (see Figure B–2). If a file extends across volume boundaries before terminating, EOV labels are written, indicating that the file continues on another volume (see Figure B–3).

Index

Reader's Comments

Guide to VMS Files and Devices
AA–LA06A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	□	□	□	□
Completeness (enough information)	□	□	□	□
Clarity (easy to understand)	□	□	□	□
Organization (structure of subject matter)	□	□	□	□
Figures (useful)	□	□	□	□
Examples (useful)	□	□	□	□
Index (ability to find topic)	□	□	□	□
Page layout (easy to find information)	□	□	□	□

I would like to see more/less ______________________________

What I like best about this manual is ______________________________

What I like least about this manual is ______________________________

I found the following errors in this manual:

Page Description

Additional comments or suggestions to improve this manual:

I am using **Version** ______ of the software this manual describes.

Name/Title ______________________________ Dept. ____________

Company ______________________________ Date ____________

Mailing Address ______________________________

______________________________ Phone ____________

-- Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

-- Do Not Tear - Fold Here

Cut Along Dotted Line

Reader's Comments

Guide to VMS Files and Devices
AA–LA06A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	□	□	□	□
Completeness (enough information)	□	□	□	□
Clarity (easy to understand)	□	□	□	□
Organization (structure of subject matter)	□	□	□	□
Figures (useful)	□	□	□	□
Examples (useful)	□	□	□	□
Index (ability to find topic)	□	□	□	□
Page layout (easy to find information)	□	□	□	□

I would like to see more/less ______________________________

____

____

What I like best about this manual is ______________________________

____

____

What I like least about this manual is ______________________________

____

____

I found the following errors in this manual:

Page Description

_____ ______________________________

_____ ______________________________

_____ ______________________________

_____ ______________________________

_____ ______________________________

Additional comments or suggestions to improve this manual:

____

____

____

____

I am using **Version** _____ of the software this manual describes.

Name/Title ______________________________ Dept. __________

Company ______________________________ Date __________

Mailing Address ______________________________

______________________________ Phone __________

-- Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

-- Do Not Tear - Fold Here

Cut Along Dotted Line