

OpenVMS Skills for Users

Instructor Guide

Training



COMPAQ

OpenVMS Skills for Users

Instructor Guide



Training

COMPAQ

Notice

The information in this publication is subject to change without notice.

COMPAQ COMPUTER CORPORATION SHALL NOT BE LIABLE FOR TECHNICAL OR EDITORIAL ERRORS OR OMISSIONS CONTAINED HEREIN, NOR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL.

This guide contains information protected by copyright. No part of this guide may be photocopied or reproduced in any form without prior written consent from Compaq Computer Corporation.

The software described in this guide is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of this agreement.

Other product names mentioned herein may be trademarks and/or registered trademarks of their respective companies.

©1999 Compaq Computer Corporation. All rights reserved. Printed in the USA.

Aero, ALPHA, ALPHA AXP, AlphaServer, AlphaStation, Armada, BackPak, COMPAQ, Compaq Insight Manager, CompaqCare logo, Counselor, DECterm, Deskpro, DIGITAL, DIGITAL logo, DIGITAL Alpha Systems, Digital Equipment Corporation, DIGITAL UNIX, DirectPlus, FASTART, Himalaya, InfoPak, Integrity, LicensePak, Ministation, NetFlex, NonStop, OpenVMS, PaqFax, Presario, ProLiant, ProLinea, ProSignia, QuickBack, QuickFind, Qvision, RDF, RemotePak, RomPak, ServerNet, SERVICenter, SmartQ, SmartStart, SmartStation, SolutionPak, SpeedPak, StorageWorks, Systempro/LT, Tandem, TechPak, TruCluster, and Tru64 UNIX are registered in United States Patent and Trademark Office.

Atalla, C-Series, Expand, FOX, Guardian, iTP, Measure, Netelligent, and PointView are trademarks of Compaq Computer Corporation.

Microsoft, Windows, and Windows NT are registered trademarks of Microsoft Corporation.

Intel, Pentium, and Intel Inside are registered trademarks and Xeon is a trademark of Intel Corporation.

UNIX is a trademark in the US and other countries, licensed exclusively through X-Open Company Ltd.

OpenVMS Skills for Users

Instructor Guide
December 1999

Overview

Introduction	1
Course Description	2
Prerequisites	3
Resources	3
Course Organization	4
Course Goals	5
Course Nongoes	5
Course Conventions.....	6
Course Materials.....	7
Introduction.....	7
Preparing to Teach.....	8
Resources for Study.....	8
General Teaching Suggestions	8
Preparing the System	9
Selecting Equipment.....	9
Creating Student Accounts.....	9
Suggested Teaching Schedule.....	9

Module 1 — Logging In and Out of an OpenVMS System

About This Module	2
Introduction	2
Objectives.....	2
Resources	2
Gaining Access to an OpenVMS System	3
Overview	3
Local Access through a Directly Connected Terminal	4
Text-based Login.....	5
Example: Successful Login.....	6
Example: Unsuccessful Login.....	6
Graphical Login.....	8
DECwindows or DECwindows Motif Login.....	8
Common Desktop Environment Login.....	9
Network Access through Dialup	10
Network Access through a Serial Line and the Point-to-Point Protocol Utility (PPPD)	10
Example: PPPD Login	11
Example Notes: PPPD Login Example	11
Network Login through a Terminal Server.....	12
Overview	12
Example: Login through a Terminal Server.....	13
Example Notes: Login through a Terminal Server.....	13

Terminal Server Commands.....	14
SHOW SERVICES Command.....	14
Example: List of Services	14
Example Notes: List of Services	14
Connecting to a Service	14
Example: Connecting to a Service Successfully	14
Example: Connecting to a Service Unsuccessfully	15
Displaying a List of Active Sessions	15
Example: SHOW SESSIONS Command	15
Disconnection during a Session	16
Example: Disconnected Session.....	16
When Reconnection Works.....	16
Multiple Sessions.....	17
How many Sessions may I have?	17
How to Create Another Session	17
How Sessions are Organized.....	17
Moving Between Sessions	17
Example: Multiple Sessions	18
Example Notes: Multiple Sessions	19
The Forward Switch Character.....	21
Example: Changing Sessions	21
Duration of the Forward Switch Character	21
Example: Using a Forward Switch Character	22
Example Notes: Using a Forward Switch Character	22
Deleting a Forward Switch Character.....	23
Network Login through the OpenVMS SET HOST Command.....	23
Example: SET HOST command	23
Example Notes: SET HOST Command.....	23
Capturing Screen Output of Remote Sessions	23
How the System Validates a User	24
Identifying Users	24
User Names	24
Passwords.....	25
Successful Validation	26
Unsuccessful Validation	27
Using Simple User Security Techniques	28
Overview	28
Changing a User Password.....	28
Password Screening by the System	28
Reasons for Changing a Password	28
Guidelines for Selecting a Password	29
How to Set a User Password.....	30
Example: Using the SET PASSWORD Command.....	30
Example: Possible Errors in Setting a Password.....	31
Example Notes: Possible Errors in Setting a Password.....	32
Example: Password about to Expire	32
Example: Password Expired	32
Example: System-Generated Passwords.....	33

Example Notes: System-Generated Passwords.....	33
Locking a Terminal Session on a Terminal Server	34
Making Sure Your Terminal Is Secure	34
Example: Locking Your Terminal.....	34
Example Notes: Locking Your Terminal.....	34
Locking a DECwindows Session	34
Locking a Common Desktop Environment Session.....	34
Terminating an OpenVMS Session	35
Overview	35
Logging Out of OpenVMS.....	36
LOGOUT Command	36
Example: Displaying Accounting Information at Logout	36
Example: Logging Out of a Terminal Server.....	36
Exiting DECwindows	37
Exiting the Common Desktop Environment	37
Exiting a PPP Session.....	37
Exiting a Telnet Session.....	37
Summary	38
Concepts.....	38
Gaining Access to an OpenVMS System	38
Accessing a Remote OpenVMS System.....	38
Using Simple User Security Techniques	38
Terminating an OpenVMS Session	38
Commands.....	39
Gaining Access to an OpenVMS System	39
Accessing a Remote OpenVMS System.....	39
Using Simple User Security Techniques	40
Terminating an OpenVMS Session	40

Module 2 — Issuing Commands

About This Module.....	2
Introduction	2
Objectives.....	2
Resources	2
Constructing a DCL Command.....	3
Overview	3
Constructing DCL Commands	4
DCL Commands.....	4
Components of a DCL Command	4
Figure: DCL Command Structure	5
Figure Notes: DCL Command Structure	5
Rules for Constructing a DCL Command.....	6
Modifying DCL Commands.....	7
Types of Qualifiers	7
Command Qualifiers.....	7
Positional Qualifiers	7
Using DCL to Interact with the System.....	8
Overview	8

Entering DCL Commands.....	9
Long Command Lines	9
Abbreviating DCL Commands.....	9
Comment Character	10
How the Comment Character Works.....	10
Understanding System Messages	11
Error Messages	11
Figure: Format of a System Message	12
Figure Notes: Format of a System Message	12
Severity Levels	13
HELP/MESSAGE Utility.....	14
Using DCL Features	15
Overview	15
DCL Command Line Prompting	16
Example: DCL Prompting	16
Using Wildcards	17
File Specification Wildcards	17
Uses for Wildcards	17
Using the Control Key	18
Control Key	18
Sample Control Key Sequences	18
Effect of Ctrl/O on Command Execution	18
Recalling DCL Commands	19
RECALL Command	19
Example: Displaying All Commands in the Recall Buffer.....	19
Example Notes: Displaying All Commands in the Recall Buffer.....	19
RECALL/PAGE Command	19
Recalling and Entering a Command.....	20
Up and Down Arrow Keys.....	20
Editing Command Lines	21
Entering Commands	21
Editing Commands	21
Enabling Command Line Editing.....	22
Two Editing Modes	22
Selecting an Editing Mode.....	22
Control Keys Used in Command Line Editing	23
Example: Command Line Editing Session	24
Combining DCL Commands.....	25
Using PIPE to Combine DCL Commands.....	25
Format of the PIPE Command	25
Examples of the PIPE Command	25
Summary	26
Concepts.....	26
Constructing a DCL Command.....	26
Using DCL to Interact with the System.....	26
Using DCL Features	26
Commands.....	27

Module 3 — Naming and Storing Files

About This Module	2
Introduction	2
Objectives	2
Resources	2
Constructing an OpenVMS File Specification	3
Overview	3
OpenVMS File Specifications	4
File Specification	4
Format of a File Specification	4
Parts of a File Specification	4
Sample File Specifications	5
Guidelines for Naming Files	5
Creating and Identifying Subdirectories	6
Overview	6
OpenVMS Directory Structure	7
System and User Directories	7
Figure: Directory Structure	7
Figure Notes: Directory Structure	8
Maximum Number of Subdirectory Levels	8
Creating Subdirectories	9
Identifying Subdirectories	9
Rules for Naming Subdirectories	9
Creating a Subdirectory	9
Example: Creating a Subdirectory	10
Common Error in Creating a Subdirectory	10
Using Subdirectories	11
Uses for Subdirectories	11
Displaying Your Default Device and Directory	11
Navigating a Directory Structure	11
Specifying a Higher Directory Level	12
Example: Using SET and SHOW DEFAULT	12
Example Notes: Using SET and SHOW DEFAULT	13
Common Error: Setting Default to a Nonexistent Directory	13
Common Error: Using an Incomplete Subdirectory Name	13
Specifying Many Directory Levels	15
Displaying Information About Files	16
Overview	16
DIRECTORY Command	17
Listing Files	17
Example: DIRECTORY Display	17
Controlling the DIRECTORY Display	18
Displaying Detailed Directory Information	18
Example: Displaying Detailed Directory Information	19
Writing a Directory Listing to a File	19
Using Wildcards	20
Wildcard Characters	20

Example: Using Wildcard Characters	20
Summary	21
Concepts.....	21
Constructing an OpenVMS File Specification.....	21
Creating and Identifying Subdirectories	21
Displaying Information About Files.....	21
Commands.....	22
Creating and Identifying Subdirectories	22
Displaying Information About Files.....	22

Module 4 — Creating Memos, Reports, and Data Files

About This Module	2
Introduction	2
Objectives.....	2
Resources	2
Performing Basic EVE Operations.....	3
Overview	3
Invoking EVE.....	4
EDIT/TPU Command	4
Editing a New File	4
To edit an existing file:	4
Figure: EVE Screen Layout	5
Figure Notes: EVE Screen Layout	5
Entering EVE Commands	6
Predefined Keys.....	6
Figure: VT200 Keypad Definitions.....	7
Figure Notes: VT200 Keypad Definitions	7
Typing EVE Commands	8
Moving the Cursor.....	9
Cursor Movement Keys	9
Cursor Movement Commands.....	9
Creating and Modifying a Text File	10
Overview	10
Entering Text.....	11
General Text Entry Pattern	11
Two Modes of Text Entry	11
Replacing Text	12
REPLACE Command	12
Replacement Options.....	13
Requesting Help with EVE	14
Overview	14
EVE Online Help.....	15
How to Request Help.....	15
Kinds of Help Topics.....	15
Help Key	15
Using Some Advanced Features of EVE	16
Overview	16
Including Existing Files into a File.....	17

INCLUDE FILE Command	17
Example: Including a File	17
Executing DCL Commands from EVE	18
Entering a DCL Command from the EVE Command Line	18
SPAWN Command.....	18
Creating Alternate Key Definitions	19
Defining Your Keypad.....	19
Simulating the EDT Keypad	19
Simulating the WPS Keypad.....	20
Saving an Edited File.....	21
Overview	21
Ending a Session.....	22
Two Ways to End a Session.....	22
Quitting a Session.....	22
Exiting from a Session.....	22
Saving a File Without Ending Your Editing Session	23
Two Ways to Save a File	23
SAVE FILE Command.....	23
SAVE FILE AS Command.....	23
File Recovery.....	24
Journal File.....	24
Types of Journaling	24
How to Recover a Lost Editing Session	25
Summary	26
Concepts.....	26
Performing Basic EVE Operations.....	26
Creating and Modifying a Text File	26
Requesting Help with EVE	26
Using Some Advanced Features of EVE	26
Saving an Edited File.....	26
Commands.....	27
Performing Basic EVE Operations.....	27
Creating and Modifying a Text File	27
Requesting Help with EVE	27
Using Some Advanced Features of EVE	27
Saving an Edited File.....	28

Module 5 — Manipulating Files

About This Module.....	2
Introduction	2
Objectives.....	2
Resources	2
Making Copies of Files.....	3
Overview	3
COPY Command.....	4
Copying Files	4
Format of the COPY Command.....	4
Requirement for the COPY Command.....	4

Sample COPY Operations and Commands	5
Modifying the COPY Command.....	6
Qualifiers.....	6
Requesting Confirmation of the COPY Operation.....	6
Example: Copy Operation Using /CONFIRM.....	7
Example Notes: Copy Operation Using /CONFIRM	7
Displaying a Log of the COPY Operation	7
Example: Copy Operation Using /LOG.....	8
Example Notes: Copy Operation Using /LOG.....	8
Setting a New Protection Code on a Copied File	8
Example: Copy Operation Using /PROTECTION	8
Common Errors in Copying Files.....	9
Errors in Copying Files.....	9
Example: Error in Input File	9
Example: Error in Output File.....	9
Displaying the Contents of Files	10
Overview	10
TYPE Command.....	11
Displaying the Contents of a File	11
Format of the TYPE Command.....	11
Defaults for the TYPE Command	11
Format of the TYPE Command.....	11
Example: Using the TYPE Command	11
Requirements and Restrictions for the TYPE Command	12
Requirement for the TYPE Command.....	12
What Kinds of Files Can be Displayed.....	12
Files that Should Not be Typed.....	12
What to Do if You Display a Non-ASCII File.....	12
Different Ways to Use the TYPE Command	13
Displaying Multiple Files.....	13
Displaying One Screen at a Time	13
Displaying a File in a Different Location	14
Printing Files	15
Overview	15
PRINT Command.....	16
Printing Files	16
Format of the PRINT Command.....	16
Requirement for the PRINT Command	16
Entry Number	16
Modifying the Action of the PRINT Command.....	17
Qualifiers for the PRINT Command.....	17
Specifying a Queue.....	17
Specifying the Number of Copies Wanted	18
Asking the System to Notify on Completion	18
Displaying Information About Print Jobs	19
Monitoring Print Jobs	19
SHOW ENTRY Command.....	19
Example: Listing Queue Entries.....	19

Example: Displaying a Specific Queue Entry.....	19
Example: Displaying Detailed Queue Information	20
SHOW QUEUE Command.....	21
Example: Displaying Queue Information	21
Controlling Print Jobs	22
Controlling Print Jobs	22
Changing the Characteristics of Your Print Job	22
Job Characteristics You Can Change	22
Example: Changing Characteristics of a Print Job.....	23
Example: Holding and Releasing a Job	23
Removing Your Print Job from the Queue	24
Requirements for Deleting a Job.....	24
Example: Removing a Print Job from the Queue.....	25
Common Errors in Manipulating Print Jobs	26
Errors in Manipulating Print Jobs.....	26
Error: Nonexistent File	26
Error: Nonexistent Queue	26
Error: Nonexistent Job.....	27
Error: Nonexistent Entry.....	27
Renaming Files.....	28
RENAME Command.....	29
Renaming a File.....	29
Format of the RENAME Command	29
Requirements for the RENAME Command	29
Sample Rename Operations	29
Using Version Numbers with the RENAME Command	30
Modifying the RENAME Command.....	31
Qualifiers.....	31
Requesting Confirmation of the Rename Operation	31
Example: Rename Operation Using /CONFIRM.....	31
Displaying a Log of a Rename Operation.....	32
Example: Rename Operation Using /LOG	32
Common Errors in Renaming Files	33
Errors in Renaming Files	33
Example: Error in Input File	33
Example: Error in Output File.....	33
Removing Files from Disk	34
Overview	34
DELETE Command.....	35
Deleting Files	35
Format of the DELETE Command.....	35
Specifying Version Numbers	35
Requirements for the DELETE Command	35
Modifying the DELETE Command.....	36
Qualifiers.....	36
Requesting Confirmation of a Delete Operation.....	36
Requesting a Log of a Delete Operation.....	36
Common Errors in Deleting Files.....	37

Errors in Deleting Files.....	37
No Version Number Specified	37
Trying to Delete a Protected File.....	37
PURGE Command.....	38
Purging Files	38
Format of the PURGE Command.....	38
Requirement for the PURGE Command.....	38
Modifying the PURGE Command	39
Qualifiers.....	39
Requesting Confirmation of the Purge Operation.....	39
Example: Purge Operation Using /CONFIRM	39
Displaying a Log of the Purge Operation	40
Example: Purge Operation Using /LOG	40
Keeping More than One Version of a File.....	40
Example: Purge Operation Using /KEEP	41
Example: Combining PURGE Command Qualifiers	41
Common Errors in Purging Files.....	42
Errors in Purging Files.....	42
Example: Specifying a Version Number When Purging Files.....	42
Example: Trying to Purge a Protected File.....	42
Example: Trying to Purge a File in a Protected Directory.....	43
Erasing the Contents of Deleted Files.....	44
Contents of Deleted Files.....	44
Erasing Files.....	44
Summary	45
Concepts.....	45
Making Copies of Files.....	45
Displaying the Contents of Files	45
Printing Files	45
Renaming Files.....	46
Removing Files from Disk.....	46
Commands.....	47
Making Copies of Files.....	47
Displaying the Contents of Files	47
Printing Files	47
Renaming Files.....	47
Removing Files from Disk.....	47

Module 6 — Communicating with Other Users

About This Module.....	2
Introduction	2
Objectives.....	2
Resources	2
Using the Mail Utility	3
Overview	3
Getting Started with the Mail Utility	4
Overview	4
Invoking the Mail Utility	4

Mail Folders	4
Figure: Default Mail Folders.....	6
Example: Identifying Your Current Folder.....	6
Exiting from the Mail Utility	6
Displaying Information About Messages	7
Listing Mail Messages	7
Example: Listing Mail Messages	7
Example Notes: Listing Mail Messages	7
Displaying a List of All Your Folders	8
Specifying a Folder.....	8
Reading Messages	9
Notification About New Messages.....	9
Disabling Notification About New Messages.....	9
READ Command.....	10
Sample Read Operations.....	10
Controlling Display of Screen Information.....	11
Example: Reading a Mail Message	12
Example Notes: Reading a Mail Message	12
Sending Messages.....	13
Ways to Send a Message.....	13
SEND Command.....	13
Editing Your Message	14
Example: Sending a Mail Message	14
Example Notes: Sending a Mail Message.....	14
Sending a File as a Message.....	15
Copying People on Your Messages.....	15
Using a Distribution List to Send a Message	15
FORWARD Command.....	16
Example: Forwarding a Mail Message	16
Example Notes: Forwarding a Mail Message	16
Using a Signature File.....	17
REPLY Command.....	17
Example: Reading and Replying to a Mail Message.....	18
Example Notes: Reading and Replying to a Mail Message.....	18
Modifying the Action of the REPLY Command.....	19
Deleting Messages.....	20
DELETE Command.....	20
How a Deleted Message Is Handled.....	20
Sample DELETE Commands.....	20
Retrieving a Deleted Message.....	20
Example: Deleting and Retrieving a Mail Message	21
Example Notes: Deleting and Retrieving a Mail Message	21
PURGE Command	22
Additional Mail Features	23
Other Features	23
Multipurpose Internet Mail Extension (MIME) Utility.....	24
Purpose.....	24
Summary	25

Concepts.....	25
Using the Mail Utility	25
Commands.....	26
Using the Mail Utility	26
Using the MIME Utility.....	27

Module 7 — Getting Help

About This Module.....	2
Introduction	2
Objectives.....	2
Resources	2
Using Online Help	3
Overview	3
OpenVMS Help Facility	4
Overview	4
Selecting Topics in HELP	4
Ending a Help Session	4
Example: Help Session	5
Example Notes: Help Session	6
Selecting a Topic from the DCL Command Line	7
Faster Use of Help	7
Help Message Utility	8
HELP/MESSAGE Command	8
Using Status Codes.....	8
Resources	8
Using the OpenVMS Documentation Set	9
Overview	9
Components of OpenVMS Documentation	10
Three Sets of Documentation.....	10
Who Should Use Each Set of Documentation	10
Documentation for OpenVMS Alpha and OpenVMS VAX.....	11
OpenVMS Master Index.....	11
OpenVMS Glossary.....	11
Summary	12
Concepts.....	12
Using Online Help.....	12
Using the OpenVMS Documentation Set.....	12
Commands.....	13
Using Online Help.....	13

Module 8 — Customizing Your Working Environment

About This Module.....	2
Introduction	2
Objectives.....	2
Resources	2
Using File Protection	3
Overview	3
Security Profiles	4

Protected Objects.....	4
Contents of a User Security Profile	4
Contents of an Object's Security Profile	4
User Identification Codes, UICs.....	5
Purpose of UICs	5
UIC Components.....	5
UIC Formats.....	5
Example: Displaying Your UIC.....	6
Example: Displaying the UIC of a File	6
Protection Codes.....	7
Types of Access.....	7
Types of File Access.....	7
User Categories	7
Assigning UICs to Categories	8
Figure: Elements of a Protection Code	9
Figure Notes: Elements of a Protection Code	9
Displaying File Security Information	10
Example: Displaying File Protection.....	11
Default Protection.....	12
Example: Displaying Default Protection	12
Example: Changing Default Protection	12
Example: Effect of Changing Default Protection.....	13
Example Notes: Effect of Changing Default Protection.....	13
Manipulating Protection Codes	14
Setting File Protection	14
Example: Setting File Protection.....	14
Protection on Directory Files	14
Steps in Deleting a Directory File	15
Example: Deleting a Subdirectory.....	15
Example Notes: Deleting a Subdirectory.....	16
Common Errors in Deleting Directory Files.....	16
Common Error: Deleting a Directory That Is Not Empty	16
Common Error: Deleting a Protected Directory File.....	16
Saving Copies of Files	18
Overview	18
Backup Utility	19
What the Backup Utility Can Do	19
BACKUP Command	19
Save Sets	19
Example: Making a Save Set of All Files in a Directory.....	20
Example: Listing the Contents of a Save Set	20
Restoring a File from a Save Set.....	21
Summary	22
Concepts.....	22
Using File Protection	22
Saving Copies of Files	22
Commands.....	23
Using File Protection	23

Saving Copies of Files	23
------------------------------	----

Customizing Your Working Environment

About This Module	2
Introduction	2
Objectives	2
Resources	2
Using Logical Names	3
Overview	3
Creating Logical Names	4
Commands Used to Create Logical Names	4
Rules for Creating Logical Names	4
Logical Names in Equivalence Strings	4
Duration of a Logical Name	5
Using Logical Names	6
Common Uses for Logical Names	6
Example: Using a Logical Name for a Directory Specification	6
Example: Using a Logical Name for a Complete File Specification	6
Example: Using a Logical Name in the Left Part of File Specification	7
Common Error in Using Logical Names	7
Using Logical Names with the Mail Utility	8
Determining the Equivalence of a Logical Name	9
Translation of Logical Names	9
SHOW TRANSLATION Command	9
Example: SHOW TRANSLATION Command	9
SHOW LOGICAL Command	9
Example: SHOW LOGICAL Command	10
Removing Logical Names	11
DEASSIGN Command	11
Delete All Process Logical Names	11
Logical Names the System Creates for You	12
Logical Names Created by the System	12
SYS\$LOGIN	12
SYS\$LOGIN_DEVICE	12
Using Symbols	13
Overview	13
Introduction to Symbols	14
Symbols	14
Local Symbols	14
Global Symbols	14
Duration of Symbols	14
Data Types for Symbols	15
Creating Symbols	16
Equating Symbols to Character Strings	16
Sample String Assignments	16
Equating Symbols to Integers	16
Creating a Symbol as a Command Synonym	17
Creating Symbols that Can Be Abbreviated	17

Displaying Symbols.....	18
Sample SHOW SYMBOL Commands.....	18
Deleting Symbols.....	18
DELETE/SYMBOL Command	18
Using Keypad Definitions.....	19
Overview	19
Creating Keypad Definitions.....	20
Keys that Can Be Defined.....	20
Defining Keys.....	20
Scope and Duration of Keypad Definitions	20
Tailoring Keypad Definitions.....	21
Example: Enabling and Defining a Key	21
Example Notes: Enabling and Defining a Key	21
Common Errors in Defining Keys.....	21
Displaying Keypad Definitions	22
Displaying Keypad Definitions.....	22
Example: Displaying a Specific Keypad Definition.....	22
Example: Displaying Detailed Keypad Definitions	22
Example: Displaying All Keypad Definitions	22
Deleting Keypad Definitions.....	23
Deleting a Single Keypad Definition.....	23
Deleting All Keypad Definitions.....	23
Writing Command Procedures	24
Overview	24
Developing Command Procedures	25
Steps for Developing Command Procedures	25
Rules for Developing Command Procedures.....	25
Guidelines for Developing Command Procedures.....	26
Example: Command Procedure.....	26
Executing a Command Procedure	27
How a Command Procedure Is Executed	27
Example: Executing a Command Procedure	27
Testing Your Command Procedure.....	27
Example: Using SET VERIFY	28
Example Notes: Using SET VERIFY.....	28
Halting Execution.....	29
Executing a Command Procedure in Batch	29
Writing a Login Command Procedure.....	30
LOGIN.COM	30
Bypassing LOGIN.COM	30
Example: LOGIN.COM.....	31
Example Notes: LOGIN.COM.....	32
Common Errors in Login Command Procedures.....	33
Summary	34
Concepts.....	34
Using Logical Names.....	34
Using Symbols	34
Using Keypad Definitions.....	34

Writing Command Procedures	34
Commands.....	35
Using Logical Names.....	35
Using Symbols	35
Using Keypad Definitions.....	35
Writing Command Procedures	36

Learning Check

Learning Check Answers

Introduction

This About This Course section describes the contents of the course and suggests ways to use the material most effectively.

The following topics are discussed here:

- Course description
- Prerequisites
- Resources
- Course organization
- Course goals
- Course nongoals
- Course conventions
- Materials
- Preparing to teach
- Preparing the system

Course Description

OpenVMS Skills for Users introduces the novice to the features of elementary DCL usage.

This course covers basic and elementary skills that a novice needs to know to become a functioning user on an OpenVMS system, including:

- Logging in and out of an OpenVMS system
- Setting up terminal characteristics
- Issuing DCL commands
- Creating and editing text files
- Sending messages to other system users

Writing a simple command procedure



Note

The release of OpenVMS Alpha Version 1.5 and OpenVMS VAX Version 6.0 marked a significant milestone in the evolution of the VMS operating system. The operating system is now available on two hardware platforms for the development of applications that can run in both environments. Users can move from one platform to the other with little or no perceived difference in operations.

This course teaches standard DCL commands; students may find that their own systems have been tailored by their system manager to support the local environment. A system manager might decide:

- To use a different command language interpreter
- To change the default action of some standard DCL commands so that they do not reflect what is described in this course
- To disable some DCL commands
- To alter some system defaults, such as the DCL prompt

Prerequisites

Since this course is for novices, there are no prerequisites.

Resources

Students should have access to the following manuals to perform the recommended learning activities of this course.

- *Extensible Versatile Editor Reference Manual*
- *OpenVMS DCL Dictionary*
- *OpenVMS Glossary*
- *OpenVMS Command Definition, Librarian, and Message Utilities Manual*
- *OpenVMS System Management Utilities Reference Manual*
- *OpenVMS System Manager's Manual*
- *OpenVMS User's Manual*

A copy of the OpenVMS base documentation kit should be available for reference.

Course Organization

This course is organized into a series of modules. Each module has its own learning objectives and covers a single topic or group of closely related topics. A module consists of:

- An **introduction** describing the purpose of the module that provides motivation for mastering its objectives and outlines its contents
- One or more **objectives**, which identify the skills taught in the module. Objectives are designed to focus the student's study efforts on a selected number of skills.
- The module **resources**, which list individual reference documents that support the information in the module.
- The module **text** that consists of:
 - Descriptive text organized by topic
 - Illustrations that clarify the relationships among various elements of an OpenVMS system or summarize steps of a particular process or command
 - Examples containing sample listings from actual interactive sessions on an OpenVMS system
- A module **summary** that reviews important concepts and commands taught in the module

Written and laboratory exercises are also provided as a separate book with this course. These exercises help students to review and practice the skills learned during the lecture session.

Course Goals

To become a functioning user on an OpenVMS system a student should be able to:

- Log in and out of an OpenVMS system
- Construct and enter DCL commands
- Construct and use OpenVMS file specifications
- Describe and move around directories
- Create and edit text files using the EVE editor
- Use DCL commands to manipulate and find files
- Use the Mail utility to communicate with other users
- Use the Help utility and the OpenVMS documentation set to get help about the OpenVMS system
- Describe and use UIC protection codes
- Customize the working environment through the use of logical names, symbols, and key definitions in a simple login command file

Course Nongoals

This course will not provide:

- Any training or experience beyond the very elementary level of the topics it covers
- Any audience-specific training that would relate to any other identifiable group of users, such as system managers, operators, or programmers
- In any of these cases, there is a user curriculum option that will provide this training.
- Training in the use of specific applications, such as spreadsheets or word processing packages

Course Conventions

This table describes the conventions used in the listings and command tables of this course.

UPPERCASE \$ TYPE OPERATOR.LOG	In commands, uppercase characters indicate words you type exactly as they appear. For example, you would type the commands on the left as they appear.
lowercase \$ TYPE file-spec	Lowercase characters represent elements you must replace according to the description in the text. For example, you must follow certain rules when you replace "file-spec" in the command on the left.
Horizontal ellipses \$ TYPE file-spec, <HELLIPSIS>	Horizontal ellipses indicate you can enter additional parameters, values, or information. For example, you can enter any number of file specifications in this TYPE command.
Vertical ellipses \$ TYPE MYFILE.DAT . . . \$	Vertical series of periods or ellipses mean that not all of the data that the system would display in response to the particular command is shown, or that not all the data a user would enter is shown.
Square Brackets ([]) \$ SHOW PROCESS [name]	Square brackets indicate the enclosed item is optional in the context of a syntax statement. For example, the process name is optional in the command shown. (Square brackets are not optional, however, when specifying directories in the syntax of a file specification.)

Course Materials

Introduction

This instructor guide has two sections. The first has general information about the course including teaching strategies, suggested schedules, and other hints. The second consists of the student materials, with additional detail and teaching notes.

Read the student information at the beginning of this About This Course section. It lists the goals and prerequisites for the course, and shows a course map.

This course consists of alternate lectures and labs and is designed to be taught in three days. Students receive a course guide and an exercise booklet.

The instruction is divided into units called **modules**. Each module covers a skill or related group of skills required to fulfill the course goals.

The students who typically attend this course have widely varying backgrounds. Some may have experience on other systems or microcomputers, some may have never seen a computer screen and keyboard before.

This course stresses only the tasks that a novice user needs to perform elementary and often repetitive tasks. The audience for this course is defined at the lowest sophistication level possible for the topics covered. However, since each class is composed of different students with different backgrounds, you should try to determine the general level of experience of the group, and customize your presentation of the material to meet their needs.

This course is presented at the correct level of difficulty for its place in the OpenVMS curriculum. No prior knowledge of OpenVMS operating system concepts is necessary to succeed in this course.

Preparing to Teach

To prepare to teach this course, you need each of the following:

- An instructor package (contains student material)
- Access to a copy of each document listed as a reference for the course

You should read each lab and written exercise and be sure you have the materials available to support them.

Resources for Study

In addition to the course materials, discussion in the Notes Conference for this course will provide you with insight into teaching these materials and give you the benefit of input from other instructors who have taught the course.

General Teaching Suggestions

The first eight modules of this course present basic information, with very few "shortcuts." The intention is to get the students used to standard methods of accomplishing tasks on an OpenVMS system. The final module of the course presents shortcuts and customization techniques, focused on LOGIN.COM.

As especially useful commands are presented, consider listing them on the side of the board, on a flipchart, or on a blank transparency. Characterize these as "useful things to do in every session". When LOGIN.COM is presented, refer to this list as suggestions for things the students might want to include in a login command file.

Many instructors find it useful to have their students try out commands as they are taught, and we encourage you to do this as well. To do this there should be at least one terminal or workstation available for every two students.

Preparing the System

Before offering this course, you must prepare a system to support the laboratory exercises that accompany the written materials. To prepare for the course, complete the following steps:

- Select a system.
- Create student accounts.

Selecting Equipment

The following equipment must be available to conduct the course. See the OpenVMS Software Product Description (SPD) for further information about supported configurations and memory requirements.

- A VAX or Alpha system running Version 7.2 or a higher version of OpenVMS software
- A terminal or workstation for each student

The exercises for this course call only for standard, nonprivileged accounts.

Creating Student Accounts

You should create a separate account for each student who enrolls in a course. Each student should also be allocated a disk quota of at least 1000 blocks if quotas are enabled. None of the operations performed by students require enhanced quotas or other special parameter values. Allow these values to default to the values specified for your training center.

Suggested Teaching Schedule

Logging In and Out of an OpenVMS System

Module 1

About This Module

Introduction

There are a variety of methods to access an OpenVMS system locally or remotely. Your system manager can identify the methods available for your use.

Once you have physically accessed the system, you must go through a procedure known as "logging in," which allows the operating system to verify that you are an authorized user. The system does this by asking you to enter identification information, which it then compares with information stored by the system manager.

This module discusses the procedures for:

- Accessing a local or remote OpenVMS system
- Logging in and out of an OpenVMS system
- How the system validates a user
- Using simple security techniques
- Terminating an OpenVMS session

Objectives

To access an OpenVMS system, a user should be able to:

- Gain access to an OpenVMS system locally and/or remotely.
- Log in to an OpenVMS system.
- Identify and use simple techniques to maintain user security.
- Terminate an OpenVMS session.

Resources

- *OpenVMS DCL Dictionary*
- *OpenVMS System Manager's Manual*
- *OpenVMS User's Manual*
- *OpenVMS Guide to System Security*

Gaining Access to an OpenVMS System

Overview

To use an OpenVMS system requires two steps – accessing the system and logging in. Accessing the system may be done locally or remotely over the network.

To use an OpenVMS system requires you to connect to the system, and then to authenticate yourself to that system. Connecting to the system may be done locally or remotely over a network.

A local connection to the system can be done either by a text based terminal which is directly connected to the system, or by a graphics terminal running the Common Desktop Environment (CDE).

An OpenVMS system can be accessed over the network using dialup, a serial line connection and Point-to-Point Protocol (PPP), a Terminal Server, an OpenVMS SET HOST command, or a Transmission Control Protocol Internet Protocol (TCP/IP) utility such as Telnet.

Logging in consists of getting the OpenVMS system's attention and identifying yourself as an authorized user.

This section discusses:

- Local OpenVMS access
 - Text-based
 - DECwindows
 - Common Desktop Environment (CDE)
- Remote OpenVMS access
 - Dialup
 - Serial line and Point-to-Point Protocol utility (PPPD)
 - Terminal Server
 - OpenVMS SET HOST command
 - TCP/IP Telnet command
- Logging in
- How the system validates a user

Local Access through a Directly Connected Terminal

When OpenVMS starts, it determines the type of each directly connected terminal. Based on the capability of that terminal, you are presented with either a text-based or a graphical login screen.

Depending on your DECwindows version, your OpenVMS platform, and your local preferences and settings, your graphical login will be through DECwindows, through DECwindows Motif, or the Common Desktop Environment (CDE) interface in DECwindows.

Before attempting either type of login, you need to obtain your user name and password(s) from your system manager.



Note

The graphical requirements for DECwindows and CDE are the same, but DECwindows was replaced by CDE in version 7.0.

If the terminal is set up as a Secure Server (\$ SET TERM/SECURE/PERMANTENT), the Break key is REQUIRED to initiate a session.

Text-based Login

If your terminal is directly connected to the OpenVMS system you follow these steps to login:

1. Obtain your username and password(s) from your system manager.
2. Turn the terminal's power on.
3. Press the **[Return]** key and you will see your system's welcome message.
4. At the Username: prompt, type your user name and press **[Return]**.
5. In response to the Password: prompt, type your password and then press **[Return]**. The system does not display your password for security reasons.

You may be informed that your password has expired and be given an opportunity to reset it. You will be asked to type it in twice to avoid typographical errors. It is a good idea to select a password with numbers, underscores, or dollar signs in it. Passwords are discussed in more detail later in this module.

The first time you login to the OpenVMS system, you may be informed that your password has expired. You will then be asked to specify a new password. You will need to type it twice to avoid typographical errors. Passwords are discussed in more detail later in this module.



Note

On some systems you may be asked to provide a secondary password for security purposes. Secondary passwords are intended for two-user logins, where two users are required to be present to log in such as in situations where any single user cannot be trusted – having one user know both passwords is not an expected use of a secondary password.

Example: Successful Login

The following example shows a successful login:

```
Username: USER1
Password:
Welcome to OpenVMS™ Alpha Operating System Version 7.2
  Last interactive login on Friday, 22-JAN-1999 11:07:33.21
  Last non-interactive login on Friday, 22-JAN-1999
08:50:47.05
$
```

The format for displaying CPU time and elapsed time is: D HH:MM:SS.CC (days, hours, minutes, seconds, hundredths of seconds)

Example: Unsuccessful Login

The following example shows an unsuccessful login:

```
Username: USER1
Password:
User authorization failure
```

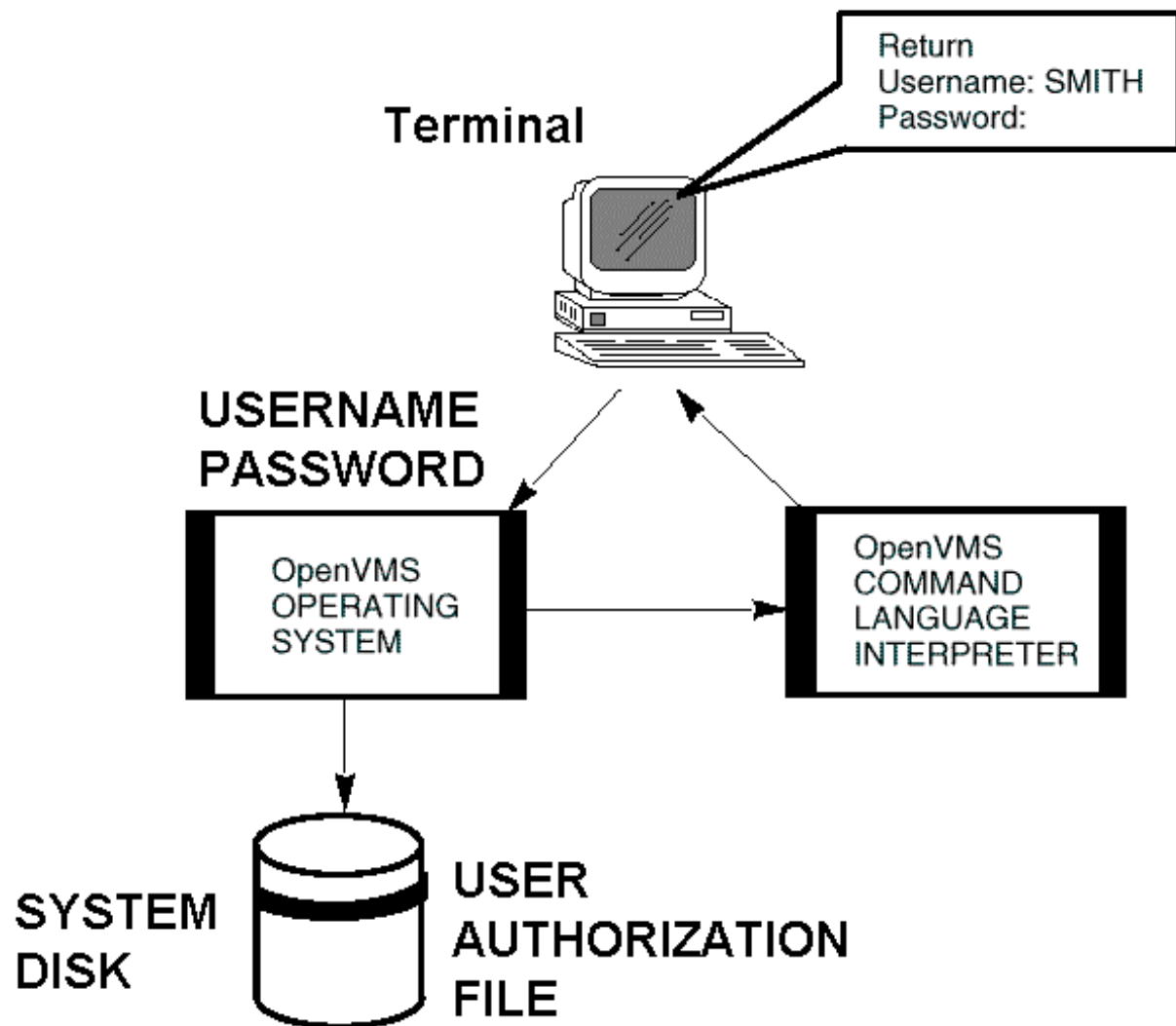


Figure: Pictorial View of Login

Figure Notes: Pictorial View of Login

- At the directly connected terminal, the user presses [Return] and enters a user name and password in response to the prompts.
- The user's password is not echoed on the screen.
- User name and password are verified against the system security database which may be on the local system or accessed over the network on a remote system.
- If they match, control is passed to DCL which displays the \$ prompt.

Graphical Login

If you are presented with a graphical login screen when you power on your directly connected terminal, you are using either OpenVMS's DECwindows interface, DECwindows Motif, or The Common Desktop Environment. These provide both a command line and a graphical user interface (GUI).

DECwindows or DECwindows Motif Login

Follow these steps to login:

1. Obtain your username and password(s) from your system manager.
2. Turn the monitor's power on. The dialog box will welcome you to the system.
3. If necessary, click your mouse button on the dialog box. In response to the "Please enter your user name" prompt, type your user name and press [Return] or click OK.
4. In response to the "Please enter your password" prompt, type your password and then press [Return] or click OK. The system does not display your password for security reasons.
5. You may be informed that your password has expired and be given an opportunity to reset it. You will be asked to type it in twice to avoid typographical errors. If so, it is a good idea to select a password with numbers in it. Passwords are discussed in more detail later in this module.
6. On some systems you may be asked to provide a secondary password for security reasons.
7. A Session Manager or DECterm window will appear on the screen with the DCL \$ prompt.

Common Desktop Environment Login

1. Obtain your username and password from your system manager.
2. Turn the monitor's power on. The dialog box will welcome you to the system.
3. If necessary, click your mouse button on the dialog box. In response to the "Please enter your user name" prompt, type your user name and press [Return] or click OK.
4. In response to the "Please enter your password" prompt, type your password and then press [Return] or click OK. The system does not display your password for security reasons.
5. You may be informed that your password has expired and be given an opportunity to reset it. You will be asked to type it in twice to avoid typographical errors. If so, it is a good idea to select a password with numbers in it. Passwords are discussed in more detail later in this module.
6. On some systems you may be asked to provide a secondary password for security purposes.
7. You may see a dialog box that says "Starting the New Desktop for OpenVMS".
8. When you click on the triangle above the editor icon on the front panel at the bottom of your screen, you will be presented with a list of selections called a subpanel. Click on the terminal and a DECterm window will appear on the screen. When you click in that screen, the DCL \$ prompt will appear. These DECterm windows are numbered and you may open more than one DECterm window.

Network Access through Dialup

Modem access may be provided to your OpenVMS system. See your system manager for details on how to access your OpenVMS system through a modem. Once you have modem access to your OpenVMS system, follow the instructions for local login earlier in this section.

Network Access through a Serial Line and the Point-to-Point Protocol Utility (PPPD)

Use the Point-to-Point Protocol Utility (PPPD) to initiate an Internet Protocol (IP) network connection over an asynchronous, serial data line. With PPPD you may establish permanent, low-speed network connections between local hosts, such as between a laptop computer and an Alpha workstation connected by a serial data line.

The following example shows how to establish a Point-to-Point connection over a serial line.

Example: PPPD Login

You must first determine the device names of your serial connections using the DCL SHOW DEVICES command:

```
$ SHOW DEVICES
.
.
.
Device      Device      Error
Name        Status      Count
TTA0:       Online      0
TTB0:       Online      0
.
.
.
```

Then you establish the connection:

❶\$ PPPD

❷PPPD> CONNECT TTB0:

%PPPD-I-CONNECTTERM, converting connection on device _TTB0: to a Point-to-Point connection

Example Notes: PPPD Login Example

- ❶ The PPPD command invokes the PPPD utility. The CONNECT command creates a temporary network connection to the serial port TTB0. Follow instructions for local login earlier in this section.
- ❷ The port is ready to receive the PPP setup negotiations initiated by the host system at the other end of the serial connection.

Network Login through a Terminal Server

Overview

Your terminal may not be connected directly to a computer system. It might be connected to a device called a *terminal server* or communications server.

A terminal server is a hardware device with accompanying software programs that connects terminals, printers and computers in a local area *network (LAN)*. Working with a terminal server inserts another layer of communication between the user and the OpenVMS system being accessed. Terminal servers are intelligent devices with their own CLI. Terminal servers prompt the user for input with the default Local> prompt. This default prompt will be used throughout this document.

Use terminal server commands to configure and control system access. Basic commands are discussed in this section but more advanced capability may be provided by your terminal server. For more information on terminal server commands see your terminal server's documentation.

This section discusses the following:

- CONNECT command
- SHOW SERVICES command
- SHOW SESSIONS command
- Use of the **[Break]** key
- Controlling multiple sessions
- FORWARD and RESUME commands
- Using a forward switch character

Example: Login through a Terminal Server

The following demonstrates login through a terminal server:

```
❶ DECserver 500 Terminal Server V2.2 - LAT VX.X
   Please type HELP if you need assistance
❷ Enter username> DANA
❸ Local> CONNECT CD
❹ Local -010- Session 1 to CD on node NEAT established
❺ Username: USER1
   Password:
Welcome to OpenVMS™ Alpha Operating System Version 7.2 on node
NEAT
.
.
.
```

Example Notes: Login through a Terminal Server

- ❶ Press **[Return]** to communicate with the terminal server.
- ❶ Help is available after you enter your username.
- ❷ Enter your user name or any character(s).
- ❸ Connect to a computer system called CD (Course Development).
- ❸ Help is available at the Local> prompt.
- ❹ A terminal session is established on the terminal server NEAT.
- ❺ Login as discussed earlier in this section.

Terminal Server Commands

SHOW SERVICES Command

The SHOW SERVICES command displays a list of services available through your terminal server. It lists the systems available for login.

Example: List of Services

```
❶ Local> SHOW SERVICES

❷ Service Name  Status      Identification

    CD  2 Connected  OpenVMS Course Development
      NEAT Available  NEAT - Unauthorized access prohibited
    TIDY 1 Connected  TIDY - Unauthorized access prohibited
      .
      .
      .
```

Example Notes: List of Services

- ❶ At the terminal server prompt, enter the SHOW SERVICES command
- ❷ The service CD has two sessions connected to it.

Connecting to a Service

Use the CONNECT command to connect to a service. When a connection is complete, you receive the OpenVMS login prompt.

Example: Connecting to a Service Successfully

```
Local> CONNECT CD
Local -010- Session 1 to CD on node NEAT established

CD (NEAT) (VAX 7000-630)
Unauthorized access prohibited.
Username: USER1
.
.
.
```

Example: Connecting to a Service Unsuccessfully

If a connection is not completed, the server returns a message:

```
Local> CONNECT ADC
Local -711- Service not known
Local -013- Continuing attempts to connect to ADC
```

To return to the server prompt press the **[Break]** key for your terminal.

Protected services, such as those that offer dial-out modems, can require you to enter a password before the connection is allowed.

Displaying a List of Active Sessions

The SHOW SESSIONS command displays the following information about currently established sessions:

- Port number and user name
- Current session number (most recently accessed)
- Status of sessions
- Service name
- Node name if different from the service name

Example: SHOW SESSIONS Command

```
Local> SHOW SESSIONS
Port 82: K    Local mode Current session: 1
- Session 1: Connected Interactive CD    (NEAT)
```

Disconnection during a Session

Sometimes a terminal session may be disconnected. This may happen in two ways – either the login process on the OpenVMS system will stop running, or it will enter a paused state. For example, if the OpenVMS system stops running, then the login process on that system is terminated. If the reason for disconnection has to do with the terminal server or the network, the OpenVMS operating system may pause the user process until the user can reconnect to it.

To reconnect to a disconnected session:

1. CONNECT to the service that was disconnected.
2. Log in to the same user account.
3. The OpenVMS operating system will display a message saying that the user has a disconnected process and you will be given the opportunity to reconnect to it.

Example: Disconnected Session

```
Username: USER1
Password:
You have the following disconnected process:
Terminal Process name Image name
VTA227: USER1      (none)
Connect to above listed process [YES]: Y
$
```

When Reconnection Works

Reconnection only works if your system manager has enabled virtual terminals on your OpenVMS system and the terminal you are using has the Disconnect characteristic. For more information, contact your system manager. The interval where a reconnection can occur is configured by the system manager, and defaults to fifteen minutes. If no reconnection occurs within the specified interval, the process is deleted.

Multiple Sessions

You may establish multiple connections to multiple services at the same time. Multiple terminal sessions enable you to:

- Work on more than one computer system from the same terminal
- Perform more than one task at a time from a single terminal

You may have multiple sessions with different services, and/or multiple sessions with one service.

How many Sessions may I have?

The terminal server software configuration determines the number of sessions you may have at one time. Your system or network manager may change this setting. If you try to exceed your session limit, you will receive an error message, and you must disconnect a session before you can create a new one.

Additionally, you may use only one session at a time. If you are working in one session and have a task running in a different session that is trying to display output to the screen, that task will stop because it cannot display to the screen. As soon as you reconnect to that session, the display will continue.

How to Create Another Session

To establish another session:

1. Return to local mode by pressing the **[Break]** key.
2. Use the CONNECT command with the service name.

How Sessions are Organized

As you create more sessions, they are assigned numbers. The list of your sessions is circular. For example, if there are two sessions, session 2 follows session 1 and then session 1 follows session 2.

1
2
1
2 and so on

Moving Between Sessions

If you would like to skip to the next higher numbered session, type FORWARD at the Local> prompt. To return to the next lower numbered session, type RESUME at the Local> prompt.

Example: Multiple Sessions

The following example shows how a user may establish and use multiple sessions:

❶ DECserver 500 Terminal Server V2.2 - LAT VX.X

Please type HELP if you need assistance

Enter username> K

Local> CONNECT CD

❷ Local -010- Session 1 to CD on node NEAT established

CD (NEAT) (VAX 7000-630)

Unauthorized access prohibited.

❸ Username: USER1

Password:

Welcome to OpenVMS Alpha Operating System Version V7.2 on node
NEAT

Last interactive login on Friday, 20-JAN-1999 12:15:22.07

Last non-interactive login on Friday, 20-JAN-1999 09:37:51.44

\$

❹ Local> C CD

Local -101- 1 other session(s) connected

Local -010- Session 2 to CD on node TIDY established

❺ Username: USER1

Password:

Welcome to OpenVMS Alpha Operating System Version V7.2 on node
TIDY

.

.

.

\$

❻ Local> SHOW SESSIONS

Port 16: K Local mode Current session: 2

- Session 1: Connected Interactive CD (NEAT)

- Session 2: Connected Interactive CD (TIDY)

❼ Local> RESUME 2

Local -012- CD session 2 resumed

\$

❽ Local> FORWARD

Local -012- CD session 1 resumed

❾ \$ LOGOUT

USER1 logged out at 14-AUG-1995 12:41:04.69

Local -218- Connection to CD terminated

Service user disconnect request

⑩ Local> SHOW SESSIONS

```
Port 16: K      Local mode Current session: None
- Session 2: Connected Interactive CD      (TIDY)
```

⑪ Local> RESUME

```
Local -102- No other sessions active
Local -012- CD session 2 resumed
```

⑫ \$ LOGOUT

```
USER1 logged out at 20-JAN-1999 12:45:51.80
Local -218- Connection to CD terminated
Service user disconnect request
```

⑬ Local> LOGOUT

```
Local -020- Logged out port 16 on server Local
```

Example Notes: Multiple Sessions

- ① Log in to the terminal server and connect to the CD service.
- ② Terminal session number 1 is established on the system called NEAT.
- ③ Login to NEAT.
- ④ Return to the terminal server Local> prompt by pressing the **[Break]** key. Connect to the CD service again.
 - You may abbreviate commands as long as they remain unique (C = CONNECT).
 - This time the terminal server connects you to the system called TIDY.
- ⑤ Login to TIDY.
- ⑥ Press **[Break]** to return to the terminal server, and request a list of your terminal sessions. Note that they are on the same service, but on different systems.
- ⑦ Return to session 2.
- ⑧ Return to the terminal server by pressing **[Break]**, and go forward to the next session in your list of sessions. (Remember the list is circular.)
- ⑨ Log out of session 1 and you return to the terminal server prompt.
- ⑩ Only session 2 remains.
- ⑪ You are returned to the only active session.
- ⑫ Logout of session 2.
- ⑬ Logout of the terminal server.



Note

A good habit when switching between sessions is to do something like CTRL/T right after you've switched, because it tells you which system you've switched to, and the program you're currently running.

The Forward Switch Character

A forward switch character saves you time when you switch back and forth between sessions. It allows you to associate the commands used to connect to a session with a single character. By typing this character at the terminal server prompt, you quickly switch to the other session.

Example: Changing Sessions

Without using the forward switch character, you could switch back and forth as follows:

```
$ [Break]
Local> RESUME 1
```

The SET FORWARD command enables you to set a character you could type in your current session that would automatically switch you to the next higher numbered session. For example, you might set the Z character to be the forward switch character. If session 1 is on NEAT and session 2 is on TIDY, then you could access session 2 while in session 1 by typing Z. The forward switch character is not echoed to the screen.

Be careful selecting your forward switch character. Try to pick an obscure forward switch character - one you are not likely to type during a regular login session.

Duration of the Forward Switch Character

This character is not "remembered" between terminal server sessions. If you logout of the terminal server, you will have to enter the SET FORWARD command again the next time you login to the terminal server.

Example: Using a Forward Switch Character

```
❶ DECserver 500 Terminal Server V2.2 - LAT V5.1

    Please type HELP if you need assistance
    Enter username> K
❷ Local> SET FORWARD Z
❸ Local> CONNECT CD
    Local -010- Session 1 to CD on node NEAT established

❹ Username: USER1
    Password:
    Welcome to OpenVMS Alpha Operating System Version E7.0 on node NEAT
    .
    .
    .
    $
❺ Local> C CD
    Local -101- 1 other session(s) connected
    Local -010- Session 2 to CD on node TIDY established
❻ Username: SYSTEM
    Password:
    Welcome to OpenVMS Alpha Operating System Version E7.0 on node TIDY
    .
    .
    .
    $
❼ Local -012- CD session 1 resumed
    $
```

Example Notes: Using a Forward Switch Character

- ❶ Login to the terminal server.
- ❷ Set your forward switch character to be the **Z** character.
- ❸ Connect to the CD service. Terminal session number 1 is established on the system called NEAT.
- ❹ Login to NEAT.
- ❺ Return to the terminal server prompt by pressing **[Break]**, and connect to the CD service again. Session 2 is established on system TIDY.
- ❻ Login to TIDY.
- ❼ Enter **Z** (your forward switch character) to switch to session 1 without returning to the terminal server. The forward switch character is not displayed on the screen.

Deleting a Forward Switch Character

To delete a forward switch character:

```
Local> SET FORWARD NONE
```

Network Login through the OpenVMS SET HOST Command

Once you are logged in to an OpenVMS system, you may log in to another system that is linked to it through a computer network. You may also login to your system more than once using the SET HOST command.

Example: SET HOST command

```
❶ $ SET HOST TIDY
Welcome to OpenVMS Alpha Operating System Version 7.2 on node TIDY
❷ Username: SYSTEM
Password:
Welcome to OpenVMS Alpha Operating System Version 7.2 on node TIDY
  Last interactive login on Friday, 15-JAN-1999 09:44:03.99
  Last non-interactive login on Friday, 15-JAN-1999 09:46:29.20
$
❸ $ SHOW TIME
    20-JAN-1999 13:20:52

❹ $ LOGOUT
SYSTEM  logged out at 20-JAN-1999 13:21:06.11
❺ %REM-S-END, control returned to node _NEAT::
$
```

Example Notes: SET HOST Command

- ❶ Issue the SET HOST command to log in to a remote system. The message displayed may vary on your system.
- ❷ Enter your user name and password for the remote system.
- ❸ Issue DCL commands on the remote system.
- ❹ Log out of the session on the remote system.
- ❺ Control is returned to the original system.

Capturing Screen Output of Remote Sessions

Use the SET HOST/LOG command to produce a log file of a remote terminal session. This command creates a file that echoes all commands and responses to commands issued until the remote session is logged out.

There are several reasons why a user might want to keep a log of remote sessions.

- Capture output from a program, command procedure, or command.
- Make examples to use as a reminder, for showing to a colleague, and for showing to the support people when they ask, "What did you do?"

How the System Validates a User

Identifying Users

When an account for a new user is set up, the system manager must supply a user name and a password. The operating system will use these two pieces of information to identify people trying to gain access.

There are multiple scenarios for verifying user names and passwords depending on the version of OpenVMS you are using and whether External Authentication is being used.

With External Authentication users may log in at the OpenVMS prompt using their external user IDs and passwords. The PATHWORKS layered product supports this functionality. For more information, see your system manager.

User Names

OpenVMS user names:

- Consist of 1 to 12 characters
- Recommend using the same user name on OpenVMS and Windows NT.
- User names are not case sensitive.
- Often assigned to be the user's last name or the user's first initial and last name.
- Must be unique on a system

If there were two Smiths on a system, the system could not tell which one was logging in. In this case, the system manager could make the names unique by adding the first letter of the first name, L_SMITH.

Passwords

OpenVMS passwords:

- Should consist of 6 to 10 characters. Adequate length makes passwords more secure. You may choose a password as long as 32 characters.
- Are composed of the following legal characters:

A through Z (upper- or lowercase, the system will translate them to uppercase)

Numerals 0 through 9

Dollar sign (\$)

Hyphen (-)

Underscore (_)

- Usually may be changed by the user at any time
- Do not echo on the screen as the user types



NOTE

When External Authentication is used:

- Upper and lower case letters are not interchangeable.
 - User names are case sensitive.
 - Passwords are limited to 14 characters (the Windows limit).
-

Passwords are protected so they cannot be accidentally read or displayed on any terminal.

OpenVMS does not display passwords as they are typed to ensure security. It is recommended that you refrain from writing passwords down to keep them more secure.

Successful Validation

For locally authorized log in without enhanced security features, the user name and password data is maintained in a file called the User Authorization File (UAF). If the user name and password entered match those in the UAF, the system:



NOTE

In Symmetric Multi Processing, multiple CPUs can share a copy of OpenVMS and all can run in Kernel mode simultaneously.

So, users may log in to a different CPU in an OpenVMS cluster each time they log in.

1. Creates an interactive process for the user

Process is the term that an OpenVMS system uses to describe the entity that is created for a user when that user logs in. All work that is done on an OpenVMS system is done inside a process.

2. Displays an informational system message, if there is one

The system manager may create a text file containing a message or messages that will be displayed at the user's terminal at each login. These are very useful when making announcements that all users need to see.

3. Displays the default DCL prompt (\$)

At this time, control of communication is passed to the *Command Language Interpreter* (CLI). The CLI acts as the go-between between the user and the computer; it translates DCL commands into a form that the computer understands.

Unsuccessful Validation

If the user name and password do not match the authentication database:

1. The system displays the following error message:

```
User authorization failure
```

2. The user is denied access to the system.

The user may press **[Return]** to redisplay the `Username` prompt and may try again. Because typing errors may occur, OpenVMS allows you to attempt to log in more than once.

Using Simple User Security Techniques

Overview

You should protect your account against unauthorized access. You should change your password frequently and take precautions if you must leave a terminal that is logged in.

This section discusses:

- Changing a user password
- Locking a terminal session

Changing a User Password

Password Screening by the System

The OpenVMS operating system automatically screens user-generated passwords for acceptability. The system uses several methods to decide if a selected password is allowable.

- The system keeps a list of previous passwords for each user and checks that the new password is not on this list. By default, the system keeps each user's old passwords for one year.
- A new password is automatically checked against a system dictionary to make sure a good password is chosen.

Reasons for Changing a Password

There are many reasons why a user may have to change a password:

- Passwords should be changed periodically for security reasons.

Every user has a responsibility to contribute to a secure computing environment. The simplest way that users may do this is to periodically change their password. This lessens the chance that some outside person could obtain a password and use it to gain access to a system.

- The system manager may force periodic changes in passwords.



NOTE

This is done by placing a password expiration date in the UAF.

- System managers may do this on a regular basis or occasionally when they receive information that would indicate a threat to security.
- A newly created user account will usually have a pre-expired password, allowing the user to login only once using that password.

When a system manager creates a new account, both the system manager and the user know the password to the account. It is standard practice for the system manager to pre-expire the password to a new account to ensure that the user will immediately change the password. This means that only the user will know the password.

Guidelines for Selecting a Password

Follow these guidelines when selecting a password:

- Make passwords at least six characters long.

It takes longer for word-generating programs to formulate all possible combinations of letters for long words than for short ones.

- Include both numbers and letters in the password.

Although a 6-character password that contains only letters is fairly secure, a 6-character password with both letters and numbers is much more secure.

- Do not select passwords from a dictionary in your native language.
- Avoid names or words that are readily associated with you or your computer site.

Do not, for example, use your pet's name, the name of a friend or relative, or a product name. You may use multiple word phrases as a single word.

```
this_is_a_multi_word_phrase  
andsoisthis
```

- Change your password at least once every month.

Do not rotate back and forth between two or three passwords. Really change it.

How to Set a User Password

Follow these steps to change your password:

1. Type the SET PASSWORD command at the \$ prompt.
2. The system will prompt for the old password. When changing your password, user input is not echoed at the terminal.
3. The system will prompt for the new password. You must enter the new password twice. If the two entries do not match, the password does not change. This eliminates possible typing errors as you entered the password.

Example: Using the SET PASSWORD Command

The following example shows the prompts displayed when you use the SET PASSWORD command. Note that the user's responses do not echo.

```
$ SET PASSWORD
Old password:
New password:
Verification:
$
```

Example: Possible Errors in Setting a Password

The following example shows five common errors that occur when changing a password.

```
$ SET PASSWORD
Old password:
New password:
❶ %SET-F-INVPWDLEN, password length must be between 15 and 32 characters;
password not changed
$
$ SET PASSWORD
Old password:
New password:
Verification:
❷ %SET-F-PWDNOTDIF, new password must be different from current password
$
$ SET PASSWORD
Old password:
New password:
Verification:
❸ %SYSTEM-F-PWDINHIS, password found in history list; please choose a new
password
$
$ SET PASSWORD
Old password:
New password:
❹ %SYSTEM-F-PWDINDIC, password found in system dictionary; please choose another
string
$
$ SET PASSWORD
Old password:
New password:
Verification:
❺ SET-I-PWDNOTVER, new password verification error; please try again
Verification:
$
```

Example Notes: Possible Errors in Setting a Password

- ❶ The new password is shorter than the minimum length required by this site.
- ❷ The new password is the same as the current password.
- ❸ The password has been previously used.
- ❹ The password tried is in the system dictionary.
- ❺ The new password and the verification do not match.

Example: Password about to Expire

If a password is about to expire, users will see a display like the one shown below when they login.

```
Welcome to OpenVMS™ Alpha Operating System Version 7.2

Username: USER1
Password:
Welcome to OpenVMS™ Alpha Operating System Version 7.2 on node TIDY
Last interactive login on Tuesday, 19-JAN-1999 09:07:43.51
Last non-interactive login on Tuesday, 19-JAN-1999 09:09:29.17
WARNING - Your password expires on Thursday, 21-JAN-1999 11:06:21.88
$
```

Example: Password Expired

If you do not change the password before the date displayed by the system as the password expiration date, you will be allowed to log in one more time. At that time you will see a message stating that the password has expired.

```
Welcome to OpenVMS™ Alpha Operating System Version 7.2

Username: USER1
Password:
Welcome to OpenVMS™ Alpha Operating System Version 7.2 on node NEAT
Your password has expired; you must set a new password to log in
New password:
Verification:
$
```

If the user logs out without changing the password, access to the system will not be permitted until the system manager has reset or changed the password.

Example: System-Generated Passwords

Some sites may require that users use system-generated passwords. These are randomly generated combinations of letters that may be used as passwords. A user may request a system-generated password by using the SET PASSWORD/GENERATE command.

```
$ SET PASSWORD/GENERATE
❶ Old password:
❷ pyffhi
    koogaldo
    aimidder
    izuzwa
    uscoilf

❸ Choose a password from this list, or press RETURN to get a new list
❹ New password:
❺ Verification:
$
```

Example Notes: System-Generated Passwords

- ❶ When you type your current password, it does not echo on your screen.
- ❷ The system generates a list of possible passwords, with suggested pronunciations.
- ❸ You may request another list (by pressing **[Return]**) if you do not find a password you want to use.
- ❹ The new password does not echo when you type it.
- ❺ If the two passwords do not match, you will receive the following message:

```
%SET-I-PWDNOTVER, new password verification error; please try again
Verification:
```

You have three chances to verify the password before you are returned to the DCL prompt.

To cancel the command without selecting a new password, type **[Ctrl/C]** or **[Ctrl/Z]**.

Locking a Terminal Session on a Terminal Server

Making Sure Your Terminal Is Secure

If you wish to leave your terminal temporarily unattended but you do not want to logout, you may protect your session(s) from other users by issuing the LOCK command to the terminal server.

If LOCK is not enabled on your server, an error message will be displayed. If LOCK is enabled on your server, the server will prompt you for a lock password. Terminal server passwords are one to sixteen alphanumeric characters, not echoed. You must enter the password a second time for verification.

The terminal remains locked until the password is entered again, returning you to local mode.

Example: Locking Your Terminal

```
❶ Local> LOCK
❷ Lock password>
❸ Verification>
❹ Local -019- Port 16 locked
❺ Unlock password>
❻ Local>
```

Example Notes: Locking Your Terminal

- ❶ Press the **[Break]** key to enter local mode, and issue the LOCK command.
- ❷ The server prompts for a lock password (not echoed).
- ❸ You must enter the same password a second time.
- ❹ The passwords match, so a "locked" message is displayed.
- ❺ You must enter the password again to unlock the terminal.
- ❻ Entering the correct password returns you to local mode.

Locking a DECwindows Session

To lock a DECwindows session, from the session manager pull down menu, click on the pause option.

Locking a Common Desktop Environment Session

Press the padlock on the Front Panel and a dialog box will display. This dialog box will disappear eventually. Press **[Return]** to view the dialog box. Enter your password and press **[Return]** to unlock your system. The dialog box will not reflect what you type – it will remain unchanged.

Terminating an OpenVMS Session

Overview

When you finish using the system, you should log out of OpenVMS. Depending on how you have accessed your OpenVMS system, you may also need to exit your access utility. The only time you would not need to exit an access utility is if you have logged on locally to a terminal directly connected to your system. Logging out prevents unauthorized users from accessing the system and your account, and makes resources available to the system.

This section discusses:

- Logging out of OpenVMS
- Logging out of a Terminal Server
- Exiting Windows
- Exiting the Common Desktop Environment
- Exiting a Point-to-Point Protocol Utility Session
- Exiting a Telnet Session

Logging Out of OpenVMS

LOGOUT Command

Use the LOGOUT command to end your terminal session as shown below:

```
$ LOGOUT
USER1 logged out at 20-JAN-1999 11:10:32.12
```

Example: Displaying Accounting Information at Logout

If your site charges users for computer time, information displayed by the LOGOUT/FULL command may be helpful to you.

```
$ LOGOUT/FULL
USER1 logged out at 20-JAN-1999 11:10:32.12

      Accounting information:
      Buffered I/O count:    79   Peak working set size: 1560
      Direct I/O count:     26   Peak page file size:  5229
      Page faults:         5760   Mounted volumes:      0
      Charged CPU time: 0 00:00:01.95 Elapsed time:  0 00:02:48.31
```

The format for displaying CPU time and elapsed time is: D HH:MM:SS.CC (days, hours, minutes, seconds, hundredths of seconds)

Example: Logging Out of a Terminal Server

In addition to logging out of your OpenVMS session, you need to log out of the terminal server if you are connected to one. In the following example, the user logs out of the terminal session on an OpenVMS system, and then logs out of the terminal server.

```
$ LOGOUT
USER1 logged out at 20-JAN-1999 11:11:37.17
Local -011- Session 1 disconnected from CD
Local> LOGOUT
Local -020- Logged out port 16 on server Local
```

Exiting DECwindows

In addition to logging out of your OpenVMS session, you need to log out of DECwindows. Exiting DECwindows is done by closing your last DECterm window or exiting out of Session Manager. Remember to save any customizations made prior to exiting.

Exiting the Common Desktop Environment

In addition to logging out of your OpenVMS session, you need to exit out of the Common Desktop Environment. This is accomplished by clicking on the EXIT button found on the front panel. The EXIT button causes your customizations to be saved.

Exiting a PPP Session

In addition to logging out of your OpenVMS session, you need to exit PPPD. Once the serial line connection is terminated, use the PPPD EXIT command to return to the DCL \$ prompt as follows:

```
PPPD> EXIT  
$
```

Exiting a Telnet Session

In addition to logging out of your OpenVMS session you need to exit the Telnet session. When you have returned to the Telnet prompt, DISCONNECT your session and EXIT Telnet as follows:

```
TELNET> DISCONNECT  
TELNET> EXIT  
$
```

Summary

Concepts

Gaining Access to an OpenVMS System

- When you log in to an OpenVMS system, you must enter a user name and password.
- The steps for logging in are:
 - Enter your username
 - Enter your password
- You may login locally or over the network in order to access your OpenVMS system.
- If you access your system through a terminal server, you may have more than one session running at a time.

Accessing a Remote OpenVMS System

- Use the SET HOST command to log in to another system that is linked to yours through a network.
- You may keep a log of your remote session.

Using Simple User Security Techniques

- You should change your password frequently and select a password that is not easy to guess.
- Use the SET PASSWORD command to change your password.
- You may protect your session(s) from other users by locking your display when you must leave your terminal temporarily unattended.

Terminating an OpenVMS Session

- Use the LOGOUT command to end a terminal session.
- If you are connected over the network you must log out of the access method you have used.

Commands

Gaining Access to an OpenVMS System

CONNECT *service-name*

Connect to the specified service.

**DISCONNECT SESSION *session-number*
DISCONNECT ALL**

Disconnect the specified session, or all of your sessions.

SHOW SERVICES

Display information about network services currently available to your port.

SHOW SESSIONS

Display a list of your sessions, in order of their creation.

RESUME [*session-number*]

Return to the specified session, or to the current session if you do not specify a session number.

FORWARD

Return to the session following the current one in the list of sessions.

BACKWARD

Return to the session preceding the current one in the list of sessions.

SET FORWARD *switch*

Specify a switch character to use when you want to move between sessions without returning to the terminal server prompt (local mode).

Accessing a Remote OpenVMS System

SET HOST *node_name*

Establish a connection to any system available over Ethernet cables from the current system.

SET HOST/LOG *node_name*

Establish a connection to any system on the network from the current system and create a log file of all commands and system responses during the remote session.

Using Simple User Security Techniques

SET PASSWORD

Change your password.

SET PASSWORD/GENERATE

Set a new password by using a system-generated character string.

LOCK

Safely leave your terminal unattended when you have at least one active terminal server session.

Terminating an OpenVMS Session

LOGOUT

End an OpenVMS session.

Issuing Commands

Module 2

About This Module

Introduction

Once you have accessed the OpenVMS system and logged in, you have established a session. Digital Command Language (DCL) enables you to communicate with OpenVMS by entering commands. Everything that you do during a session, such as:

- Organizing and handling files
- Using an editor to prepare reports, memos, and other documents
- Using electronic mail
- Getting help while on line

may be accomplished by issuing one or more DCL commands. This module introduces you to the structure and features of DCL.



Note

This module talks about DCL commands in general. Specific commands will be discussed elsewhere, as appropriate.

Objectives

To issue DCL commands, a user should be able to:

- Identify the components of a DCL command and use them to construct error free DCL commands.
- Enter DCL commands.
- Use DCL features such as command recall, command line editing, and control key sequences to efficiently use DCL commands.

Resources

- *OpenVMS DCL Dictionary*
- *OpenVMS System Messages and Recovery Procedures Reference Manual*
- *OpenVMS User's Manual*
- *OpenVMS System Messages: Companion Guide for Help Message Users*

Constructing a DCL Command

Overview

DCL, the Digital Command Language, uses English-like words for its commands and qualifiers. This use of common words makes it easier to:

- Remember DCL commands
- Enter DCL commands at a keyboard
- Recognize and correct grammatical errors

DCL commands can be used to:

- Perform file manipulation tasks such as copying, renaming, and deleting files and directories
- Display directories and the files in them
- Display information about the system, users currently on the system, devices connected to the system, resources available to users
- Run system utilities, applications, or programs

This section discusses:

- Constructing DCL commands
- Modifying DCL commands

Constructing DCL Commands

DCL Commands

DCL commands are built in a manner similar to the way we construct sentences. There are several parts that go together in a specific order to make a command. Just as in sentences, there are also optional parts that can be added to modify the basic command.



Note

DCL locates the programs from commands entered using DCL command tables.

Components of a DCL Command

Element	Comment and Example
Verb	The verb of the command line specifies the action of your request. \$ HELP
Keyword	Sometimes a verb requires a keyword to further clarify the action \$ SHOW USERS In this example, USERS is a keyword for the verb SHOW.
Parameter	The action of the verb is taken on the parameter. Parameter values include names of files or system resources. \$ PRINT THURSDAY.TXT
Qualifier	The qualifier(s) describe or modify the action taken by the verb. A slash(/) precedes each qualifier. \$ SHOW PROCESS/ALL Note: There may or may not be spaces before the "/" character.
Qualifier value	Some qualifiers accept one or more values. A value assigns a specific quantity to the qualifier. \$ PRINT VERMONT.TXT/COPIES=2



Note

The SHOW PROCESS/ALL command displays the basic subset of information as well as information about:

- Accounting
 - Dynamic memory use
 - Privileges
 - Quotas
 - Shelving
 - Subprocesses
-

Figure: DCL Command Structure

The following figure shows the structure of a typical DCL command.

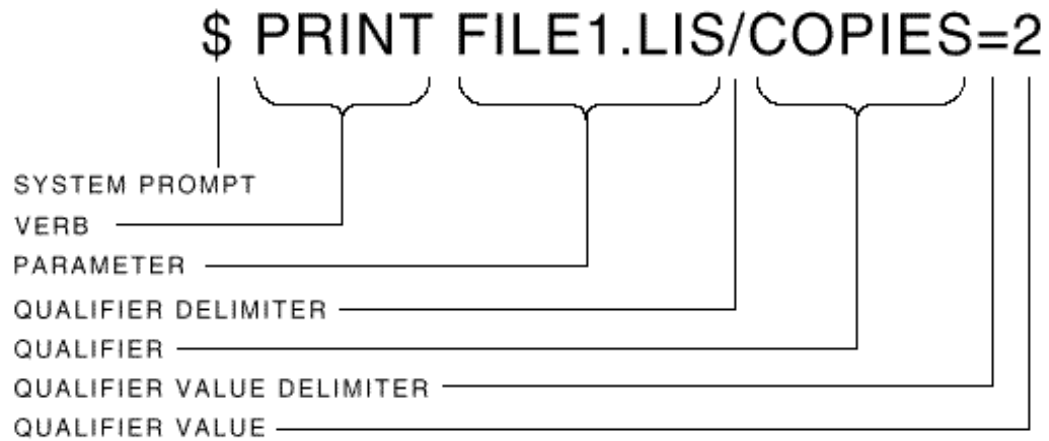


Figure Notes: DCL Command Structure

- The entire line is called the *DCL command line*.
- `$` is the default OpenVMS system prompt.
- `PRINT` is the *verb* of the DCL command.
- `FILE1.LIS` is the *parameter*; it is the file that is to be printed.
- `/COPIES=2` is a *qualifier* and a *value* that modifies the way the `PRINT` command works. It asks for two copies to be printed instead of one.
- Press **[Return]** to terminate the command line.



Note

- The system prompt displayed is actually the default system prompt and may be modified by the system manager and/or user. It may be the system name and/or current directory.
- Not all commands have parameters or qualifiers.
- A "delimiter" is a character used to separate elements of a command string, file specification, etc.

Rules for Constructing a DCL Command

Follow these grammatical rules when you construct a DCL command:

1. Use any combination of uppercase and lowercase letters.

LOGOUT and logout are interchangeable.

2. One or more spaces or tabs separate commands, qualifiers, and parameters.

DCL needs to see white space between commands, and parameters. The amount of white space does not matter.

The following two command lines are interpreted equally by DCL:

```
$ PRINT          FILE1.TXT          /COPIES=2
```

and

```
$ PRINT FILE1.TXT /COPIES=2
```

3. Slashes (/) distinguish qualifiers from commands and parameters. Begin each qualifier with a slash. It does not matter if there are spaces surrounding the slash or not.
4. These are equivalent:
/COPIES=2 and / COPIES=2
5. Commas (,) separate parameters.

```
$ PRINT FILE1.TXT, FILE2.TXT, FILE3.TXT
```

Modifying DCL Commands

Types of Qualifiers

There are two types of qualifiers for DCL commands:

- Command qualifiers
- Positional qualifiers

They are named to designate the part of the command line that they modify - either the command itself or other information such as a parameter.

Command Qualifiers

Command qualifiers mean the same no matter where they appear in the command line.

Examples:

```
$ PRINT/HOLD FILE1.TXT  
$ PRINT FILE1.TXT/HOLD
```

Since /HOLD is a command qualifier, these two commands have the same effect. Both commands place the print request in a hold state (submit the file to the printer, but don't print it now).

Positional Qualifiers

The location of the positional qualifiers determine their impact on the command.

- A positional qualifier placed after the verb, but before the first parameter, affects the entire command line.

The following command causes two copies of FILE1.TXT and two copies of FILE2.TXT to be printed.

```
$ PRINT/COPIES=2 FILE1.TXT, FILE2.TXT
```

- A positional qualifier placed after a parameter affects only that parameter.

The following command causes one copy of FILE1.TXT and two copies of FILE2.TXT to be printed.

```
$ PRINT FILE1.TXT, FILE2.TXT/COPIES=2
```

Using DCL to Interact with the System

Overview

While you are logged in to an OpenVMS system, you conduct a dialogue with the system. You communicate with the system by entering DCL commands. The system responds either by executing your command or by displaying a system message.

This section discusses:

- Entering DCL commands
- Understanding informational and error messages displayed by the system

Entering DCL Commands

To enter a DCL command:

1. Type the command with its parameters and/or qualifiers on the keyboard.
2. Press [Return]



Note

If a student has used the arrow keys to make changes, they often think they must move the cursor to the end of the command line before pressing RETURN. Stress that this is not necessary.

Long Command Lines

When entering long DCL commands, you can:

1. Continue typing if your terminal has the wrap feature set.

Use the SHOW TERMINAL command to see if your terminal has WRAP or NOWRAP set. If WRAP is set, your typing will automatically be continued on the next line.

```
$ PRINT FILE1.TXT/COPIES=3 ,  
FILE2.TXT/COPIES=2, FILE3.TXT/COPIES=4
```

(or)

2. Terminate the line with a hyphen (-) and press **[Return]**. DCL prompts you with an underscore followed by the system prompt, or what you may have changed your prompt to, (set prompt) and you may continue typing. The SET PROMPT command may be used to change your prompt.

```
$ SHOW -  
_ $ TIME
```

Abbreviating DCL Commands

Commands may be abbreviated. You must type enough characters to uniquely identify the command. If you use only the first four characters DCL will recognize the command. All DCL commands are guaranteed to be unique to four letters.

Comment Character

You may insert comments in a command line. An exclamation point indicates that DCL is to ignore all characters following it.

This may be very useful for documenting DCL command procedures (programs with DCL commands in them).

How the Comment Character Works

For trial purposes, type the following:

```
$ SHOW USERS !This command will list users on your system
```

The system will execute the SHOW USERS command which lists the current users. It will ignore the comment after the exclamation point. If you were to enter the same line without the exclamation point, you would get an error message because DCL would try to read the comment as part of the command.

Understanding System Messages

DCL issues messages in response to many of the commands you enter. The system does not usually display messages that confirm the success of a command. The return of the prompt signifies success. System messages tell if an operation was or was not successful and may suggest possible causes of the error. DCL provides two types of messages – success and error.

Error Messages

Error messages generally occur for the following reasons:

- A command is entered incorrectly. It contains a spelling or grammatical error or an incorrect parameter.
- A resource you are attempting to use is currently unavailable. Tape drives, disks, and other devices may not always be available for use.
- An error is detected when a program executes.
- A system has a software or hardware problem.

For more information about messages refer to the *OpenVMS System Messages and Recovery Procedures Reference Manual*

Figure: Format of a System Message

The following figure displays a system error message. The user has incorrectly typed THINK, which is not a DCL command.



Figure Notes: Format of a System Message

%	The system message delimiter precedes all system messages.
DCL	The facility code names the portion of OpenVMS that has detected the error.
W	The severity level code describes the severity of the error. W indicates warning.
IVVERB	The message identification code is an abbreviation of the message which may be used to locate information about that message in documentation.
Unrecognized. . .	The message text gives a brief but more detailed explanation of the message and may suggest an action.



Note

Recall that the SET MESSAGE command can be used to control what elements of the message are displayed.

Severity Levels

There are five levels of severity in a system message.

S (Success)

The command completed successfully.

```
$ COPY/LOG A.TXT B.TXT
%COPY-S-COPIED, TIDY$DKA200:[USER]A.TXT;1 copied to. . .
```

I (Informational)

DCL sometimes displays additional information regarding an operation's success.

```
$ PURGE/LOG
%PURGE-I-FILPURG, TIDY$DKA200:[USER]TEST.LIS;2. . .
%PURGE-I-FILPURG, TIDY$DKA200:[USER]TEST.LIS;1. . .
%PURGE-I-TOTAL, 2 files deleted (6 blocks)
```

W (Warning)

Part of your request may have been performed.

```
$ DIRECTORY NONEXISTENT.FILE
%DIRECT-W-NOFILES, no files found
```

E (Error)

The command is incorrect.

```
$ DELETE A.TXT
%DELETE-E-DELVER, explicit version number. . .
```

F (Fatal or Severe)

The action requested by the command failed.

```
$ DEASSIGN Y
%SYSTEM-F-NOLOGNAM, no logical name match
```



Note

- Remind students to check the severity code: not all system messages indicate problems.
 - The system messages shown above are used only for illustration. They have been truncated for convenience.
-

HELP/MESSAGE Utility

Use the Message utility (MESSAGE) to display information about how the last command completed:

```
$HELP/MESSAGE
```

You can also display information about a specific message by including the message identifier or words from the message text. For example:

```
$HELP/MESSAGE BADACP
```

For details on how to use the Help Message utility, see *OpenVMS System Messages: Companion Guide for Help Message Users*.

Using DCL Features

Overview

DCL is modeled on English vocabulary and rules of grammar. DCL provides convenient ways of entering commands.

This section discusses:

- DCL command line prompting
- Using wildcards on the DCL command line
- Using the control key
- Recalling DCL commands
- Editing command lines

DCL Command Line Prompting

Some DCL commands will prompt you for additional information. With some commands DCL provides a default parameter.

Example: DCL Prompting

Perhaps you want to make a copy of a file, but you cannot remember if the old file name or the new file name should be typed first. You may type the COPY command, press the **[Return]** key and the system will prompt you for more information.

```
$ COPY
```

```
_From: FILE1.TXT
```

```
_To: NEW_FILE.TXT
```

Using Wildcards

File Specification Wildcards

A *wildcard* is used to abbreviate. It is a special character or characters that are a shortcut for you to type. It represents additional characters. By using wildcard characters in file specifications, you can apply the action of a DCL command to more than one file at a time.

The wildcards used when naming files are:

* (asterisk)

- Matches any number of characters in the position it occupies

% (percent)

- Matches any single character in the position it occupies
- Can occur more than once
 - ♦ MY%.TXT looks for MY followed by any single character followed by .TXT
 - ♦ MY%%.TXT looks for MY followed by any two characters followed by .TXT

Uses for Wildcards

A command entered with wildcards will apply to all files that match the wildcard specification.

```
$ DELETE MY.*TXT;*
```

This command deletes all versions of all files that start with MY, are followed by any number of characters, and have a type of TXT.

You can use wildcards in the middle of a filename as well as at the end of filenames.

```
$ DELETE FILE*.DAT;
```

The following command should be used with extreme caution:

```
$ DELETE *.*;*
```

DCL will not ask if you really want to delete all the files in the current directory. It will just delete them.

Using the Control Key

Control Key

The control key is marked [Ctrl] on most keyboards and is used in combination with other keys to send special messages to the operating system.

A control sequence is issued by pressing and holding [**Ctrl**] while striking another key, much in the same manner that you capitalize a letter by pressing and holding [**Shift**] while striking another letter.

Using control sequences can save you time and keystrokes.

Sample Control Key Sequences

The following examples show how the control key and other keys may be combined to issue special commands.

Use this key combination...	To perform this function:
[Ctrl/S]	Pause scrolling your screen. This is used to read a section of output.
[Ctrl/Q]	Restart scrolling of your screen after you have stopped it with [Ctrl/S]. The display continues from the point where it was paused.
[Ctrl/O]	Stop and restart (toggle) display of output to the screen. This differs from [Ctrl/S] and [Ctrl/Q] in that it does not pause the output; it only stops output from being displayed. When you resume output to the screen by issuing another [Ctrl/O], the output will start at its current point. This is useful in skipping over large amounts of output when you are looking for something in a very large file.



Note

- Ctrl/S stops the flow of information.
 - Ctrl/Q makes the flow go quickly again.
 - Ctrl/O turns the output off and on.
-

Effect of Ctrl/O on Command Execution

Note that if you use [Ctrl/O], it takes much less time for the computer to process output from a command because it does not have to write it out to the screen. For instance, if you are displaying the contents of a very large file, it will take much less time to output several pages while [Ctrl/O] is in effect than it will to display those same pages on the terminal screen.

Recalling DCL Commands

RECALL Command

DCL allows previously executed commands to be recalled. Changes may be made and the command executed. When a user has made an error typing a lengthy command recalling it eliminates the possibility of making a different error while retyping.

The UPARROW may be used to recall commands. Once they are recalled, they may be modified and executed.

The RECALL command displays previously entered commands. Up to 255 commands are stored in the recall buffer on all OpenVMS systems.



Note

Previous RECALL commands are not retained in the RECALL buffer, and will not be displayed when you use RECALL/ALL, CTRL/B, or the UPARROW.

Example: Displaying All Commands in the Recall Buffer

Use the RECALL/ALL command to display a listing of the stored commands:

```
❶ $ RECALL/ALL
❷ 1 SHOW USERS
  2 SHOW TERMINAL
  3 SET PASSWORD
  4 DIRECTORY
$
```

Example Notes: Displaying All Commands in the Recall Buffer

- ❶ The RECALL/ALL command displays the previously entered commands.
- ❷ Each command is assigned a number in the display.

RECALL/PAGE Command

Many commands have a /PAGE qualifier which displays output one screen at a time. RECALL/PAGE is useful to display the 255 stored commands.

Recalling and Entering a Command

You may select a specific command to change and execute by either:

- Entering the RECALL *n* command, where *n* is the number of the command.
\$ RECALL 3 displays the third command

(or)
- Entering RECALL *command*
\$ RECALL SHOW displays the most recent SHOW command.

When you have modified the command, press **[Return]** to execute the command.

Up and Down Arrow Keys

The UPARROW and DOWNARROW keys may be used to recall commands.

Use this key...	To perform this function:
Up arrow or CTRL/B	Consecutively recalls the previous command. (B for back)
Down arrow or CTRL/R	Consecutively recalls the next command. (R for redisplay)

Editing Command Lines

Entering Commands

Pressing **[Return]** passes the command to DCL for execution regardless of the cursor's position on the line. **[Return]** signals DCL to read all characters on the line and interpret them as a command.

Editing Commands

Before pressing **[Return]** you may make any changes you like to the command line. Commands may be recalled and changed. Text may be added, deleted or modified. DCL does not read the command until the **[Return]** key is pressed. Remember that you do not have to position the cursor at the end of the line before pressing **[Return]**.

Enabling Command Line Editing

Use the SET TERMINAL command to enable command line editing.

SET TERMINAL/LINE_EDITING

Use the SHOW TERMINAL command to determine which type of line editing is being used.

SHOW TERMINAL

Two Editing Modes

There are two modes for editing command lines.

Use this mode...	To edit commands in this way:
Overstrike	Each character typed replaces the character at the current cursor position. Overstrike is the default mode.
Insert	Characters to the right of the cursor move right to make room for new characters.

Selecting an Editing Mode

Use the SET TERMINAL command to change your terminal's editing mode.

Enter one of the following commands:

SET TERMINAL/OVERSTRIKE

SET TERMINAL/INSERT

Control Keys Used in Command Line Editing

A variety of keys and control key sequences can be used to edit a DCL command line.

Use this key...	To perform this function:
Left arrow or Right arrow	Move the cursor to the left or right within the current command line.
[Ctrl/H] or [F12]	Move the cursor to the left end (head) of the command line.
[Ctrl/E]	Move the cursor to the right end of the command line.
[Ctrl/A] or [F14]	Toggle the keyboard between overstrike and insert modes for the current command.
[DELETE]	Delete the character to the left of the cursor.
[LF] or [Ctrl/J] or [F13]	Delete the preceding word or group of characters.
[Ctrl/U]	Delete all characters to the left of the cursor.



Note

- Ctrl/T tells about your process.
- Ctrl/H takes the cursor to the head of the line.
- Ctrl/E takes the cursor to the end of the line.

Example: Command Line Editing Session

1. You mistype a command without noticing it, and press **[Return]**.

```
$ SNOW DEFAULT
%DCL-W-IVVERB, unrecognized command verb - check validity and spelling \SNOW\
```

2. You enter another command before you notice your error.

```
$ SHOW TIME
7-FEB-1999 23:50:55
```

3. You realize the previous error, and enter RECALL/ALL to list entries in the recall buffer.

```
$ RECALL/ALL
1 SHOW TIME
2 SNOW DEFAULT
```

4. You type RECALL 2 to display the incorrectly typed command. The system displays the command line.

```
$ RECALL 2
$ SNOW DEFAULT
```

5. Now use control characters to edit the command line. Type [**<--**] to position the cursor at the N.

```
$ SNOW DEFAULT
```

6. Press the **[H]** to overstrike the N.

```
$ SHOW DEFAULT
```

7. Press **[Return]**. The cursor does not need to be positioned at the end of the line.

```
$ SHOW DEFAULT
```

```
TIDY$DKA200:[PICCADACI]
```

Combining DCL Commands

Using PIPE to Combine DCL Commands

DCL commands may be combined in a manner similar to UNIX using the PIPE command. The PIPE command executes multiple DCL commands in a single command line. Combining commands in this manner reduces the amount of time a user spends managing the overhead of storing temporary results in a file.

The PIPE command may be used to direct a command's output to a file or read input from a file.

Format of the PIPE Command

PIPE command-sequence ; command-sequence [; command sequences]...

Examples of the PIPE Command

```
$ PIPE COPY A.TXT B.TXT ; DIRECTORY
```

The COPY command is executed first. Once the file has been copied, the DIRECTORY command lists all files in the current directory. The directory listing would verify that the file had been copied.



Note

The spaces on either side of the semicolon (;) MUST be there.

```
$ PIPE DIRECTORY > DIRLIST.TXT
```

The DIRECTORY command's output is redirected to the DIRLIST.TXT file instead of being displayed. DIRLIST.TXT may be viewed with the TYPE command.

Summary

Concepts

Constructing a DCL Command

- DCL commands are put together like English sentences.
- Use command and positional qualifiers to modify the action of a DCL command.

Using DCL to Interact with the System

- DCL offers a number of helpful features, including command continuation and abbreviation.
- OpenVMS system messages tell you whether a specified operation was successful and suggest possible causes of error.

Using DCL Features

- DCL will prompt a user for missing parameters.
- Use the asterisk (*) and percent (%) wildcards to apply the action of a single DCL command to multiple files.
- The control key (**[Ctrl]**) modifies the meaning of many terminal keys.
- Use the RECALL command or the up arrow key to recall previously issued commands, either to modify them or to reenter them.
- You can edit the DCL command line using a variety of terminal keys.

Commands

RECALL

Displays the most recently entered command.

RECALL n

Displays the *nth* command stored in the recall buffer.

RECALL command

Displays the most recent iteration of the specified command in the recall buffer.

RECALL/ALL

Displays all commands stored in the recall buffer.

RECALL/PAGE

On OpenVMS Alpha systems, displays all commands stored in the recall buffer, one screen at a time.

Naming and Storing Files

Module 3

About This Module

Introduction

Almost every operation a user performs in the course of a terminal session involves the use of files. To use a file, you must be able to specify its name and location. When you have created many files, it is helpful to group them into a subdirectory structure for ease in accessing them.

This module discusses the creation and use of OpenVMS file specifications and subdirectories.

Objectives

To name and store files, a user should be able to:

- Construct an OpenVMS file specification.
- Create and identify subdirectories.
- Obtain a listing of files.

Resources

- OpenVMS DCL Dictionary
- OpenVMS User's Manual

Constructing an OpenVMS File Specification

Overview

When you create or name a file, you must specify certain information so the system can locate and identify the file. This information is called a *file specification*.

This section discusses:

- OpenVMS file specifications

OpenVMS File Specifications

File Specification

A full file specification completely describes the access path the operating system uses to locate and identify a file.

Format of a File Specification

A full file specification has the following format:

NODE::DEVICE:[DIRECTORY]FILENAME.TYPE;VERSION

Parts of a File Specification

The following table describes the parts of a file specification.

This component...	Supplies this information:
Node	What system the file is on. A maximum of six characters can be used. In OpenVMS cluster systems and networks, files do not have to be on the same computer system to be used by a command or a program.
Device	The device or disk where the file resides.
Directory	Many systems have several disks or storage areas where files are kept. The directory that contains a file. The OpenVMS operating system provides a hierarchical method of organizing a storage area on disk called directories. These directories identify locations on a specific device.
File name	The actual name of the file. Valid characters are: A through Z (treated as uppercase), 0 through 9, underscore (_), hyphen (-), and dollar sign (\$) (generally avoided because of possible confusion with names generated or used by the OpenVMS operating system). File names can have from 1 to 39 characters.
File type	Information about the contents of the file, preceded by a period (.) Many types of files have default file types that allow users to immediately identify the function of a file. Examples include: ▪ Command procedures, .COM ▪ Executable images, .EXE
Version	A number preceded by a semicolon (;) that specifies the version number of a file relative to other copies of the same file. The highest version number you can use is 32767.

Sample File Specifications

A simple file specification consists of a file name and file type.

APRIL_REPORT.TXT

A complete file specification consists of file name and file type plus other information that locates and identifies the file.

NEAT::DUA0:[SMITH]APRIL_REPORT.TXT;3

Guidelines for Naming Files

Suggested guidelines for naming files include:

- Use a file name that reflects the contents or usage of the file.
- Use standard default file types.
- Avoid the use of the dollar symbol (\$) so personal files are not easily confused with system files in which the \$ occurs in the name.
- Avoid overuse of special characters.
- Don't use a hyphen as the first or last character in a file name.

Creating and Identifying Subdirectories

Overview

Each disk contains a main directory, called the master file directory (MFD), which contains a list of user file directories (UFDs). In most cases, a UFD exists for each user on the system.

The only directory that is never a subdirectory is [000000]. Subdirectories enable users to organize files into meaningful groups. For example, a user might have one subdirectory for memos and another for command procedures.

This section discusses:

- OpenVMS directory structure
- Creating subdirectories
- Using subdirectories

OpenVMS Directory Structure

System and User Directories

The OpenVMS operating system stores information in a hierarchical structure.

- At the top of this hierarchy is the *master file directory* (MFD).
- The *user file directory* (UFD) is listed in this master file directory, along with the user file directories of other users.
- The UFD of each user is often called the *login directory* or *default directory* because this is the directory used by default when a user logs in.
- Below the UFD, the user can create personal subfile directories (SFDs) called subdirectories.

Figure: Directory Structure

The following figure shows a sample directory hierarchy.

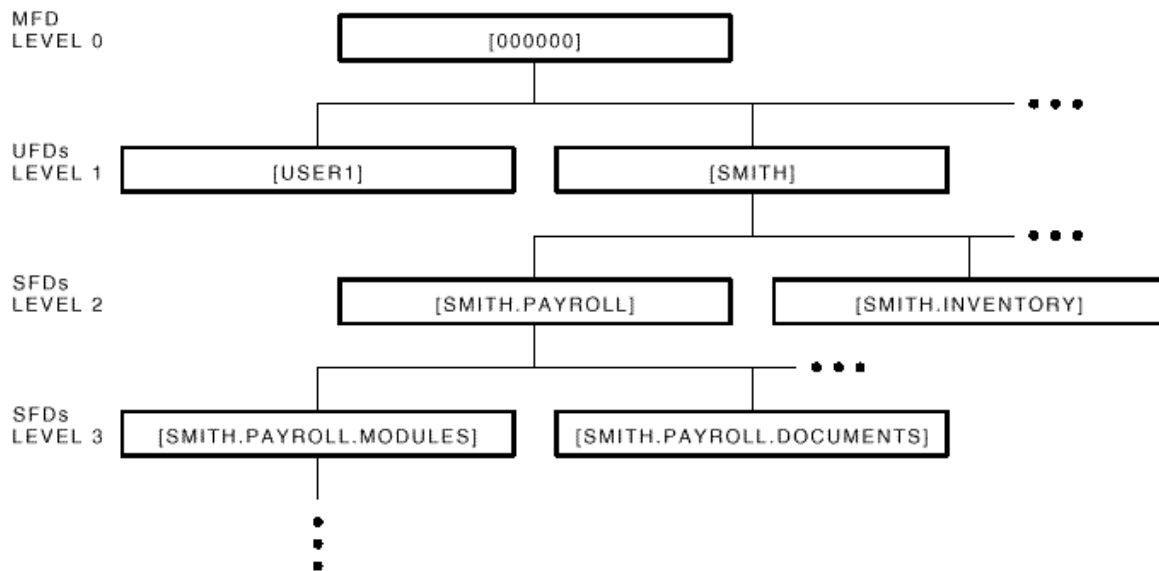


Figure Notes: Directory Structure

- MFD is the Master File Directory, which catalogs all of the user directories.
- UFD is the User File Directory.
- SFD refers to subfile directories, or subdirectories.
 - USER1 has no subdirectories.
 - SMITH has two levels of subdirectories.

Maximum Number of Subdirectory Levels

For OpenVMS V7.2 and later, each user file directory (UFD) can have a maximum of 255 levels of subdirectories below it.

Creating Subdirectories

Identifying Subdirectories

The format for directory and subdirectory specifications is:

[DIRECTORY.SUBDIRECTORY.SUBDIRECTORY...]

A period (.) between two directory names identifies the rightmost directory as a subdirectory of the directory that is named immediately to its left.

A subdirectory file has a file type of .DIR and is always assigned a version number of 1 by the system. You can identify subdirectories easily by issuing the command:

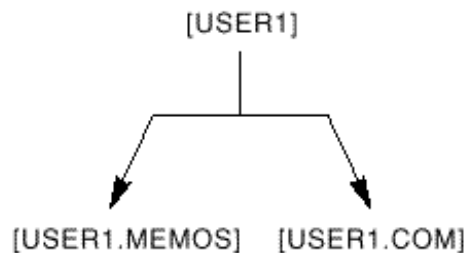
\$ DIRECTORY *.DIR

This command will list all directory files in your current directory.

Rules for Naming Subdirectories

- The subdirectory name must be enclosed in brackets.
- Use a period (.) to separate subdirectory names.
- The subdirectory name includes the directory name where it is created.

To completely name the subdirectory .MEMOS under the UFD [USER1], you



must use [USER1.MEMOS]. If you set default to [USER1], you can name the subdirectory by using [.MEMOS].

Creating a Subdirectory

Use the **CREATE/DIRECTORY** command to create a subdirectory.

Syntax:

CREATE/DIRECTORY [directory.subdirectory]

This command creates a subdirectory under the specified directory regardless of your current default directory.

Example: Creating a Subdirectory

In the following example, the optional /LOG qualifier is used to cause the system to display an informational message when the subdirectory is created. By default, the system just returns the \$ prompt to indicate that it has successfully executed the command.

```
$ CREATE/DIRECTORY/LOG [USER1.TRAVEL]
%CREATE-I-CREATED, $1$DUA1:[USER1.TRAVEL] created
$
```

Common Error in Creating a Subdirectory

A common error in creating a subdirectory is the omission of the period (.) in the subdirectory specification.

Example:

```
$ CREATE/DIRECTORY/LOG [TRAVEL]
%CREATE-E-DIRNOTCRE, [TRAVEL] directory file not created
-SYSTEM-F-NOPRIV, no privilege for attempted operation
$
```

Without the period, this command attempts to create a user file directory. To create a TRAVEL subdirectory under [USER1], use one of the following commands:

```
$ CREATE/DIRECTORY [USER1.TRAVEL]
$ CREATE/DIRECTORY [.TRAVEL]
```

Using Subdirectories

Uses for Subdirectories

Use subdirectories to:

- Organize the directory structure
- Protect files from accidental modification or loss
- Make it easier for the user to find files

Files are usually grouped by:

- Function (for example, all report files)
- Application (for example, all files for a given project)
- Type (for example, all command procedures)

Displaying Your Default Device and Directory

Use the SHOW DEFAULT command to display your current default device and directory.

Example:

```
$ SHOW DEFAULT
WORK3: [USER1]
$
```

Navigating a Directory Structure

Use the SET DEFAULT command to change the default device and/or the directory name for your process.

Syntax:

\$ SET DEFAULT *device-name*:[*directory-name*]

- A value for *device-name* does not need to be specified if the new default is on the same device.
- A physical device name must be terminated with a colon (:).
- The *directory-name* might include a subdirectory specification.
- Enclose the directory or subdirectory name in square brackets.

Specifying a Higher Directory Level

Use the hyphen character (-) to specify a directory level above your current default without specifying the upper directory name.

This command...	Performs this function:
<code>\$SET DEFAULT [-]</code>	Sets your default to the directory one level above your current default.
<code>\$ SET DEFAULT [--]</code>	Sets your default to the directory two levels above your current default.

Example: Using SET and SHOW DEFAULT

The following example shows the use of the SET DEFAULT and SHOW DEFAULT commands.

1. `$ SHOW DEFAULT`
`WORK3:[USER1]`
2. `$ SET DEFAULT [USER1.COMMANDS]`
`$ SHOW DEFAULT`
`WORK3:[USER1.COMMANDS]`
3. `$ SET DEFAULT [-]`
`$ SHOW DEFAULT`
`WORK3:[USER1]`
4. `$ SET DEFAULT [ .COMMANDS]`
`$ SHOW DEFAULT`
`WORK3:[USER1.COMMANDS]`
5. `$ SET DEFAULT WORK7:[USER2.BUILD]`
`$ SHOW DEFAULT`
`WORK7:[USER2.BUILD]`
6. `$ SET DEFAULT [- .MEMOS]`
`$ SHOW DEFAULT`
`WORK7:[USER2.MEMOS]`

Example Notes: Using SET and SHOW DEFAULT

1. Display the initial default device and directory.
2. Change the default directory to the COMMANDS subdirectory. The default device is not changed.
3. Use a hyphen to set the default directory to one directory level above the current default. The default device does not change.
4. Change the default directory to the COMMANDS subdirectory again, this time using a shortcut.

The period (.) indicates a subdirectory. Since the level above the subdirectory is not specified, it is assumed to be the current default directory.
5. Change the default device and directory.
6. Use the hyphen to reset the directory default.

Common Error: Setting Default to a Nonexistent Directory

You can set your default to a nonexistent directory. No error is returned until you attempt to access the directory.

```
$ SET DEFAULT [NONEXISTENT]
$ SHOW DEFAULT
  WORK3:[NONEXISTENT]
%DCL-I-INVDEF, WORK3:[NONEXISTENT] does not exist
$ DIRECTORY
%DIRECT-E-OPENIN, error opening WORK3:[NONEXISTENT]*.*;* as input
-RMS-E-DNF, directory not found
-SYSTEM-W-NOSUCHFILE, no such file
$
```

Common Error: Using an Incomplete Subdirectory Name

A common error in accessing a subdirectory is to use an incomplete subdirectory specification.

In the following example, the use of [.MEMOS] causes an error because the system looks for that subdirectory below the [.COMMANDS] subdirectory.

```
$ SHOW DEFAULT
  WORK3:[USER1]
$ DIRECTORY *.DIR

Directory WORK3:[USER1]

COMMANDS.DIR;1      MEMOS.DIR;1

Total of 2 files.
$ SET DEFAULT [.COMMANDS]
$ DIRECTORY

Directory WORK3:[USER1.COMMANDS]
.
```

```
.  
.  
$ SET DEFAULT [MEMOS]  
$ DIRECTORY  
%DIRECT-E-OPENIN, error opening WORK3:[USER1.COMMANDS.MEMOS]*.*;* as input  
-RMS-E-DNF, directory not found  
-SYSTEM-W-NOSUCHFILE, no such file  
$
```

To correct this error, set default using the full specification.

```
$ SET DEFAULT [USER1.MEMOS]
```

Specifying Many Directory Levels

Use an ellipsis (...) to specify many levels in a directory structure. This can be used with a variety of DCL commands, including DIRECTORY, PURGE, DELETE, and others.

This command...	Performs this task:
\$ DIRECTORY [...]	Searches the current default directory and all subdirectories below it.
\$ DIRECTORY [REPORT...]	Searches all subdirectories below the directory called REPORT.
\$ PURGE [.MEMO...]	Purges files in the MEMO subdirectory and all subdirectories below it.

Displaying Information About Files

Overview

There are many situations in which you need to display information about files. For example, you may need to:

- Locate a particular file so you can print it
- Determine the date you created a file so you know if it needs to be updated
- Find out how many files you have, and how big they are, because you are concerned about disk space
- List the names of all files with a particular file type because you have forgotten the name of the file you want

This section discusses:

- DIRECTORY command
- Using wildcards

DIRECTORY Command

Listing Files

Use the DIRECTORY command to locate a particular file or get information about files.

Syntax:

DIRECTORY [file-spec[,...]]

- The file specification determines which files will be listed.
- If you omit the file specification, you will get directory information about all of the files in your current default directory.
- If you specify more than one file, separate the file specifications with either commas or plus symbols.
- You can use wildcard characters in the directory specification, file name, file type, or version number fields of a file specification to list all files that satisfy the fields you specify.

Example: DIRECTORY Display

The default DIRECTORY display gives the file name, type, and version number of each file. The following example shows the format of the output from the DIRECTORY command.

```
$ DIRECTORY
```

```
Directory USER_DISK:[USER1]
```

```
DEBUG.LOG;3      DECTERM_ERROR.LOG;223  DECTERM_ERROR.LOG;222
DECW$TERMINAL_DEFAULT.DAT;3      DECW$TEST.PS;1      GREET.COM;1
LIBRARY.SAVE;1      LOGIN.COM;25      LOGIN.LOG;1      LOGIN.LST;1
LOGIN.SAVE;1      LWK_PERSONAL.LINKBASE;1      MAIL.MAI;2
.
.
.
Total of 73 files.
$
```

Controlling the DIRECTORY Display

You can use a variety of qualifiers to control the display of the DIRECTORY command. The following table lists some commonly used qualifiers. Refer to *the OpenVMS DCL Dictionary* for more information.

To display this information...	Use this qualifier:
Size of each file	/SIZE
Files in a specified number of columns	/COLUMNS=n
Detailed information about each file	/FULL

Displaying Detailed Directory Information

Use the DIRECTORY/FULL command to display the following file information:

- File name, type, and version number
- File identification number (FID)
- Number of blocks used and allocated
- Identification code of the file's owner
- Dates of creation, last modification, expiration, last backup, effective usage, recording on media
- File owner's user identification code (UIC)
- File organization and other file attributes
- Journaling information
- Protection information

Other DIRECTORY qualifiers display subsets of this information.



Note

An OpenVMS block is 512 bytes.

Example: Displaying Detailed Directory Information

```
$ DIRECTORY/FULL SUBMIT.COM

Directory USER_DISK:[USER1]

SUBMIT.COM;1                File ID: (135,1,0)
Size:                7/8      Owner:    [12,200]
Created:    24-AUG-1995 14:38:16.87
Revised:    14-SEP-1995 11:15:38.99 (3)
Expires:    <None specified>
Backup:     <No backup recorded>
Effective:  <None specified>
Recording:  <None specified>
File organization: Sequential
Shelved state:      Online
File attributes:    Allocation: 8, Extend: 0, Global buffer count: 0
                   No version limit
Record format:      Variable length, maximum 0 bytes, longest 122 bytes
Record attributes:  Carriage return carriage control
RMS attributes:     None
Journaling enabled: None
File protection:    System:RWED, Owner:RWED, Group:RE, World:
Access Cntrl List:  None

Total of 1 file, 7/8 blocks.
$
```

Writing a Directory Listing to a File

Use the /OUTPUT qualifier to send the output from a directory operation to a file.

Syntax:

DIRECTORY/OUTPUT=file-spec

Example:

```
$ DIRECTORY/OUTPUT=FILE.LIST
```

Using Wildcards

Wildcard Characters

You can use wildcard characters to apply a DCL command to multiple files rather than to one file at a time. The command applies to all files that match the portion of the file specification entered.

Use the asterisk (*) or a percent sign (%) as a wildcard in a directory name, file name, and file type. You can also use the asterisk, but not the percent sign, in version numbers.

Example: Using Wildcard Characters

The following command uses the asterisk character (*) to represent part of a file name.

```
$ DIRECTORY M*
```

```
Directory USER_DISK:[USER1]
```

```
MAIL.MAI;2          MERIGO.KEEP;1      MERIGO.SAVE;1      MESSAGES.DIR;1
MONITOR.SUM;1
```

```
Total of 5 files.
$
```

The following command uses the percent character to represent three characters of the file name. Only files having a four-character file name are displayed.

```
$ DIRECTORY M%%%
```

```
Directory USER_DISK:[USER1]
```

```
MAIL.MAI;2
```

```
Total of 1 file.
$
```

Summary

Concepts

Constructing an OpenVMS File Specification

- A file specification provides the system with the information it needs to identify a file.
- The parts of a file specification are:
`NODE::DEVICE:[DIRECTORY]filename.filetype;version`

Creating and Identifying Subdirectories

- The OpenVMS operating system stores information in a hierarchical structure.
- Use the `CREATE/DIRECTORY` command to create a subdirectory.
- Use subdirectories to organize your files.
- Use the `SHOW DEFAULT` command to display your current default device and directory.
- Use the `SET DEFAULT` command to change the default device and/or the directory name for your process.
- Use the ellipsis mark (...) and hyphen (-) as a shortcut in specifying directory levels.

Displaying Information About Files

- To locate a particular file or get information about files, use the `DIRECTORY` command.
- When you want information about several files with similar file names or file types, use wildcards to represent matching characters.
- Use the asterisk (*) or a percent sign (%) as a wildcard in a file specification.

Commands

Creating and Identifying Subdirectories

CREATE/DIRECTORY

Creates a directory on an OpenVMS system.

SHOW DEFAULT

Displays the current default device and directory.

SET DEFAULT

Sets a default device and directory.

Displaying Information About Files

DIRECTORY

Displays a listing of files.

Creating Memos, Reports, and Data Files

Module 4

About This Module

Introduction

One of the most common tasks for a user is the creation and modification of text files. Text files can assume a variety of formats and may serve many purposes. They may be:

- Memos and letters
- Data files that are used by programs and utilities
- Computer programs written in a language like FORTRAN

OpenVMS provides a number of ways to create, maintain, and modify text files. To create a file, you may use:

- A text editor
- The CREATE command
- The /OUTPUT qualifier of many DCL commands

This module discusses the use of EVE, the Extensible Versatile Editor, to create and modify text files.

Objectives

To create memos, reports, and data files, a user should be able to use EVE to perform the following tasks:

- Perform basic EVE operations, such as invoking the EVE editor, entering EVE commands, and moving the cursor.
- Create and modify a text file.
- Request help during an editing session.
- Use some of the advanced features of EVE such as including a file in a document, executing DCL commands from the EVE command line, and defining keys.
- Save an edited file.

Resources

- Extensible Versatile Editor Reference Manual
- OpenVMS User's Manual

Performing Basic EVE Operations

Overview

Use EVE, the Extensible Versatile Editor, to create a new file or to edit an existing file.

This section discusses:

- Invoking EVE
- Entering EVE commands
- Moving the cursor



Note

It will be helpful to have the students edit a file as you present this material.

Invoking EVE

EDIT Command

Use the EDIT command to invoke EVE.

EDIT filename.type

Editing a New File

There are two ways to create a new file in EVE.

- Enter the file name upon creation.

EDIT filename.type

(or)

- Leave the file name blank until you exit the editor and are prompted to enter a file name.

EDIT

```
.  
:  
:  
Type filename for buffer MAIN...
```

To edit an existing file:

- Supply the name of the file on the command line.

EDIT filename.type

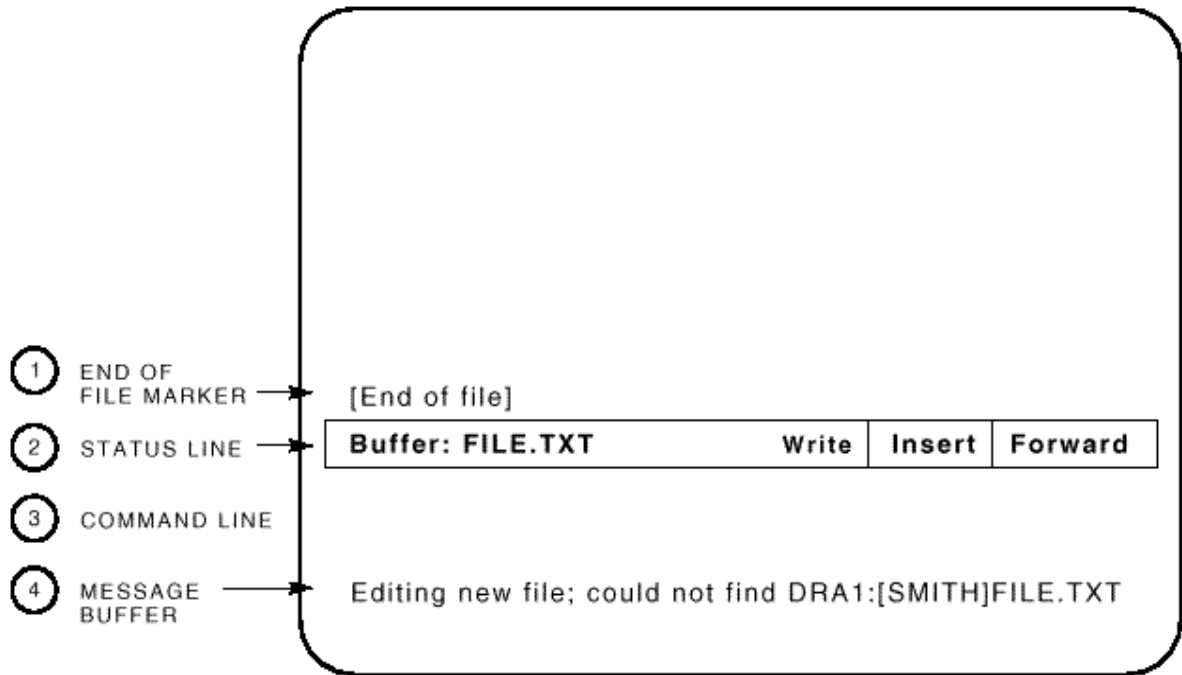
(or)

- Leave the file name blank on the command line and use the EVE INCLUDE command to include the file.

```
[Do]  
Command: INCLUDE filename.type  
.  
:  
:
```

When you exit the editor, you will be prompted for a file name.

```
Type filename for buffer MAIN (press RETURN to not write it):
```

Figure: EVE Screen Layout**Figure Notes: EVE Screen Layout**

- ❶ The [End of file] marker indicates the end of the text file.
- ❷ The status line shows:
 - The name of the buffer you are currently editing
 - The mode for entering text (the mode is Insert)
 - The direction of the cursor (the direction is Forward)
- ❸ The command line is used to enter line-mode commands. To enter line-mode commands, press **[Do]** (PF4 on some terminals and workstations) to get the Command: prompt.
- ❹ The message buffer stores EVE messages you receive during your editing session.

Entering EVE Commands

Predefined Keys

There are two ways to enter EVE commands:

- Press predefined editing keys.
- Type the commands themselves.

EVE defines some editing keys by default.

- The predefined editing keys on VT200 and VT300 series terminals include the mini-keypad, certain function keys, and certain control key sequences.
- On VT100 series terminals, EVE automatically defines most of the numeric keypad keys, the arrow keys, and certain control keys.
- Each predefined editing key performs one editing command.
- Press **[Help]** to display the keypad diagram on your screen.

Figure: VT200 Keypad Definitions

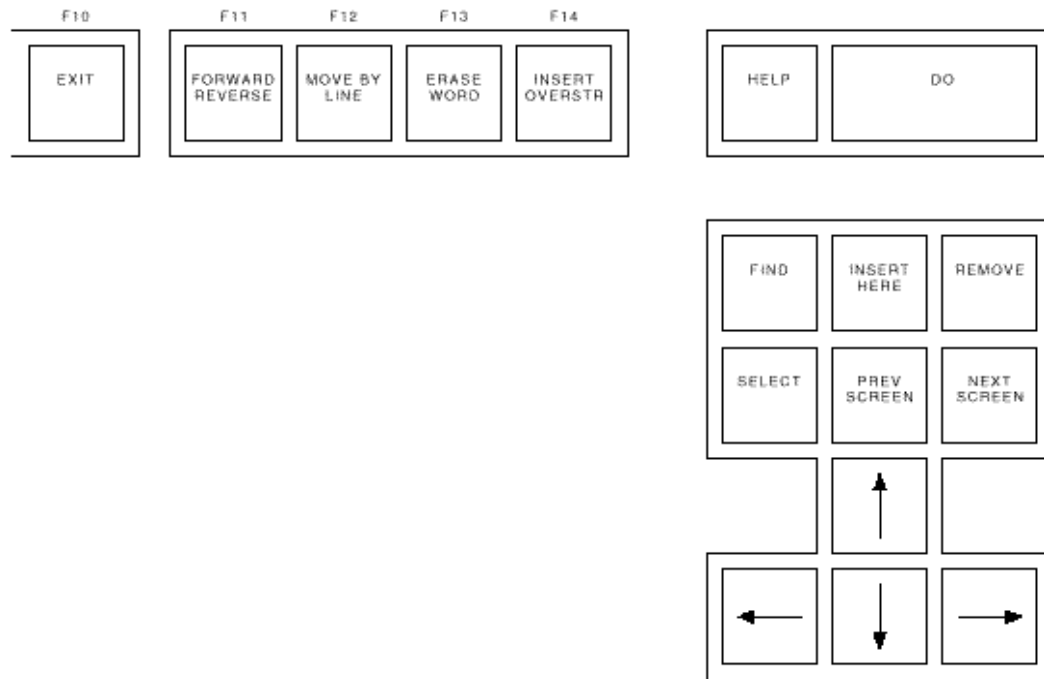


Figure Notes: VT200 Keypad Definitions

- [F11] key toggles between forward and reverse direction.
- [F12] key moves the cursor one line at a time.
- [F13] key deletes a word to the right of the cursor.
- [F14] key toggles between insert and overstrike mode.
- Press [Help] to request information about the keypad.
- Press [Do] to enter EVE commands.
- [Find] searches for a string.
- [Select], [Remove], and [Insert Here] are used for cut and paste operations.
- [Prev Screen], [Next Screen], and the arrow keys move the cursor back and forth within the file.

Typing EVE Commands

Besides using defined keys to enter EVE commands, you may type the commands. This involves three steps:

1. Press the **[Do]** key (VT300 series and VT200 series terminals) or **[PF4]** (VT100 series). The cursor will move to the command window.
2. At the Command: prompt, type the command you want to execute. You may abbreviate commands.
3. Press either the **[Do]** key or the **[Return]** key. EVE will execute the command or prompt for more information.

EVE commands may be abbreviated, as long as the abbreviation is unique.



Note

Ask students to press the Help key so they can see the keypad diagram on their screens.

Moving the Cursor

Cursor Movement Keys

EVE offers two ways to move the cursor around a text file:

- Cursor movement keys
- Cursor movement commands

Use the following keys or key combinations to move the cursor.

To move the cursor...	Use this key:
Up one line	Up arrow
Down one line	Down arrow
One character to the left	Left arrow
One character to the right	Right arrow
End of the current line	Ctrl/E
Beginning of the current line	Ctrl/H
Previous screen of the current buffer	Prev Screen
Next screen of the current buffer	Next Screen

Cursor Movement Commands

To enter any of the following commands, press the **[Do]** key, type the command, and press **[Return]**.

To move the Cursor...	Enter this command:
To the bottom of the file	BOTTOM
To the top of the file	TOP
One word in the current direction	MOVE BY WORD
One page in the current direction	MOVE BY PAGE
To the beginning of the specified line	LINE 5 (in this example, line number 5)

To request the number of the current line, enter the WHAT LINE command.

Creating and Modifying a Text File

Overview

To create and modify text files, you must be able to insert, change, and delete text in a new or existing file.

This section discusses:

- Entering text
- Replacing text

Entering Text

You may insert sections of text, entire files, and control characters into the buffer you are currently editing.

You may add entire files to the file you are editing by pressing **[Do]** and entering the EVE command `INCLUDE FILE filename.type`.

If you erase text by mistake, you can often restore the most recently erased text.

General Text Entry Pattern

Follow this pattern to enter text into your file:

- Position the cursor in the work area where you want text to appear.
- Type characters from the keyboard.

Use the **[Delete]** key to erase mistakes.

Two Modes of Text Entry

There are two modes of text entry in EVE:

Insert

- Each character you type is inserted to the left of the cursor.
- Existing text is pushed to the right.
- **[Delete]** removes the character to the left of the cursor and moves all text to the right of the deleted character one space to the left.
- This is the default setting.

Overstrike

- Each character you type eliminates a character of existing text.
- **[Delete]** replaces existing characters with a space.

Press the **[F14]** function key (or **[Ctrl/A]**) to change from one mode to the other.

Replacing Text

REPLACE Command

Use the REPLACE command to replace one character string with another.

When the operation completes, EVE displays a message indicating how many replacements were made.

To replace a character string:

1. Press the **[Do]** key.
2. Enter the REPLACE command, specifying the old and new strings.

Command: REPLACE old_string new_string

(or)

EVE will also prompt you to enter the strings.

1. Press the **[Do]** key.
2. Enter the REPLACE command and press **[Return]**.
3. Type the string to be replaced following the Old string prompt, followed by **[Return]**.

Old string: old_text

4. Type the new string at the New string prompt, followed by **[Return]**.

New String: new_text

5. EVE moves the cursor to the first occurrence of the string, and prompts:

Replace? Type Yes, No, All, Last, or Quit?

When the replacement is complete, EVE displays a message of the following form:

Replaced 2 occurrences.

Replacement Options

EVE gives you five options for the replacing strings.

Give this response...	To cause EVE to take this action:
YES	Replace the string and attempt to locate another occurrence of the string in the current direction.
NO	Do not replace the string, and attempt to locate another occurrence of the string in the current direction.
ALL	Replace the string and all other occurrences of the string in the current direction. The cursor is moved to the position where the last replacement occurred.
LAST	Replace this occurrence of the string and stop the REPLACE procedure. The cursor does not move.
QUIT	Do not replace this occurrence of the string and stop the REPLACE procedure. The cursor does not move.

Requesting Help with EVE

Overview

EVE, the Extensible Versatile Editor, has an extensive online Help facility and OpenVMS provides the Extensible Versatile Editor Reference Manual for your use.

EVE provides online help for:

- Menu items
- Commands
- Keypad functions

This section discusses:

- EVE online help

EVE Online Help

How to Request Help

To request help, press **[Do]**, type HELP, and press **[Return]**. A list of topics will be displayed.

To request help on a specific topic, press **[Do]**, type HELP *topic*, and press **[Return]**.



Note

You may abbreviate the topic name. If more than one name matches your request, EVE shows a list of matching topics so you may select the one you want.

To see a listing of all help topics, type a question mark (?).

To exit from Help, press the **[Return]** key.

Kinds of Help Topics

EVE provides help on 4 topics.

This kind of topic...	Explains:
Command	EVE commands
Keypad	EDT or WPS keys that are defined as special procedures other than as EVE commands. Type HELP at the HELP screen prompt.
Informational	Areas of operation, features of the editor, or other concepts
TPU	DECTPU built-in procedures and other topics

Help Key

You may get help about the keypad by pressing the **[Help]** key.

- EVE will build a keypad diagram on your screen.
- Press the key for which you would like help.

Using Some Advanced Features of EVE

Overview

EVE provides many advanced features for users who wish to customize their editing environment or who wish to facilitate editing by defining multiple keystroke activities into a single key.



Note

EVE is built on the DEC Text Processing utility (DECTPU), a programmable editing environment. This environment allows users to write programs that define how EVE will operate.

This section discusses:

- Including existing files into a file
- Executing DCL commands from EVE
- Creating alternate key definitions

Including Existing Files into a File

INCLUDE FILE Command

You may add the contents of another file to the file you are currently editing.

You may include only one file at a time.

You may use wildcards in the file specification. If more than one file matches your request, EVE shows a list of the matching files and recalls the INCLUDE FILE command so you may select one.



Note

Including a file does not change the name of the buffer or the output file associated with the buffer. If the file you are including contains tab characters, the tab stops of the current buffer apply. Including a file does not reformat (rewrap) the text.

Use the INCLUDE FILE command to include another file in the file you are editing.

1. Position the cursor where you want to include the file
2. Press the **[Do]** key
3. Type **INCLUDE FILE file-spec**
4. Press **[Return]**

Example: Including a File

The following command includes a file called OTHER.FILE in the current buffer:

Command: `INCLUDE FILE OTHER.FILE`

Executing DCL Commands from EVE

Entering a DCL Command from the EVE Command Line

While editing a file, you may want to execute a DCL command. For example, you may need to see a directory of your files or display your current directory. You may do this without exiting EVE.



Note

EVE creates a subprocess for executing the DCL command. When the command finishes, the cursor automatically returns where it was in the editing process.

Press **[Do]** and type DCL followed by the DCL command you wish to execute:

Command: DCL DIRECTORY *.COM

- The following message will be displayed on your screen:
Creating a DCL subprocess
- Any output from the DCL command will be shown in the lower portion of your screen.
- To remove this display from your screen, press **[Do]** and type ONE WINDOW.
- When you exit from the editor, you will see another message, indicating that the subprocess created to carry out the DCL command has been terminated.

Terminating DCL subprocess

SPAWN Command

Do not use the EVE command DCL to execute a command that invokes another application (such as MAIL) or performs screen management, which would interfere with EVE.

Instead, use the SPAWN command (Remember to use the **[Do]** key first to see the Command: prompt):

Command: SPAWN MAIL

To return to your editing session, exit from the application you spawned:

MAIL> EXIT

Creating Alternate Key Definitions

Defining Your Keypad

EVE allows you to set up your keypad to correspond to the definitions provided by another editor.

This allows you to use the capabilities of EVE with the keypad arrangement of an editor that you already know.

Simulating the EDT Keypad

Use the SET KEYPAD command to set up your keypad to behave as if you were using the EDT Editor:

Command: SET KEYPAD EDT

KP below indicates number keys on the numeric keypad. **[GOLD]** is the top left key on the numeric keypad.

SET KEYPAD EDT defines an EDT like keypad, but does not fully implement EDT. The differences are:

- **[Ctrl/Z]** makes EVE write the buffer to a file and exit to the DCL prompt.
- The **[GOLD] [KP7]** combination is defined as the **[Do]** key when the keypad is set to EDT mode.
- The **[GOLD] [KP8]** combination is defined as FILL, to reformat the currently selected text or the current paragraph. If you want this key to fill only the selected text (as in real EDT mode) redefine the key as FILL RANGE.
- EVE defines the **[Enter]** key as **[Return]**.

Note that this command does not enable you to enter EDT commands by using the **[Do]** key. You only have access to the EDT like functions by pressing the keypad keys.

The **[Help]** key will now display the EDT keypad definitions. (The **[Help]** key is **[PF2]** on some terminals or workstations.)

Simulating the WPS Keypad

You can set up your keypad to use word processing keypad definitions:

```
$ SET KEYPAD WPS
```

The **[Help]** key will now display the Word Processing System (WPS) keypad definitions.

Saving an Edited File

Overview

When you exit EVE using the QUIT command, your edits are discarded. When you exit EVE using the EXIT command, a new version of your file is created. Any previously existing versions of the files remain unchanged.

EVE has many features that allow the user to be flexible in saving copies of an edited file while providing maximum security against loss of data due to editor error or OpenVMS system interruptions.

This section discusses:

- Ending a session
- Saving a file without ending your editing session
- File recovery

Ending a Session

Two Ways to End a Session

There are two ways to end an EVE editing session:

- Quit to disregard your modifications
- Exit to write your modifications to the file

Quitting a Session

To quit without saving the modifications:

- Press **[Do]** or **[PF4]**
- Type QUIT at the Command: prompt

Exiting from a Session

To exit and save the modifications, do one of the following:

- Press **[Ctrl/Z]**.
(or)
- Press **[Do]** or **[PF4]** and type EXIT at the Command: prompt.
(or)
- Press the **[F10]** function key.

Saving a File Without Ending Your Editing Session

Two Ways to Save a File

Occasionally you might want to save a file without exiting from your editing session. For example, if you want to save your current edits, and are unsure if you want to save future edits, you can save the file and then continue editing. If you decide that you do not want to save the latest edits, you can close the file and not save them, but your previous edits will be saved.

There are two ways to save a file:

- Use the SAVE FILE command
- Use the SAVE FILE AS command

SAVE FILE Command

Writes the contents of the current buffer to the file associated with the buffer without ending the editing session. If you do not specify a file name with the SAVE FILE command, EVE prompts you for an output file specification.

SAVE FILE AS Command

Writes the contents of the current buffer to the file you specify without ending the editing session. For example, if you are editing a file named FIRST.DAT, you may save it as SECOND.TXT. This command does not change the name of the buffer. It does, however, associate the buffer with the file you name so that any subsequent SAVE FILE, WRITE FILE, or EXIT command writes the buffer to the file you named. This command requires you to supply a file specification.

File Recovery

Journal File

If a system interruption occurs or a **[Ctrl/Y]** is pressed while you are editing a file, you can usually recover your session.

EVE records the keystrokes you enter in a journal file.

- Usually the editing session ends normally, and the system deletes the journal file.
- If an interruption occurs, the journal file is saved.
- Your most recent keystrokes may not be saved.

Types of Journaling

EVE uses two types of journaling.

Buffer-change journaling (default)

- Creates a journal file for each text buffer and allows you to recover one buffer at a time.
- Works on DECwindows and character cell terminals.
- Uses a default file type of TPU\$JOURNAL.

Keystroke journaling

- Does not work on a DECwindows terminal.
- Uses a file type of TJL or TPU\$JOURNAL.

To select keystroke journaling, invoke EVE with the following command:

```
$ EDIT/JOURNAL=journalfilename filename.type
```



Note

Refer to Extensible Versatile Editor Reference Manual for details.

How to Recover a Lost Editing Session

If you are editing a file when a system crashes or if you accidentally press **[Ctrl/Y]** while editing, you may recover your edits.

You must recover an editing session at a terminal of the same type as the one used for your editing session.

EDIT/RECOVER original-file-spec

The *original-file-spec* is not the name of the journal file. When asked if you want to recover the file, type Y. If you were editing SAMPLE.TXT and pressed **[Ctrl/Y]**, you would use SAMPLE._TXT.TPU\$JOURNAL to recover, but the command you would enter is:

```
$ EDIT/RECOVER SAMPLE.TXT
```

After the recovery operation completes, the journal file is deleted.

Summary

Concepts

Performing Basic EVE Operations

- Use the EDIT command to invoke the EVE editor.
- Enter EVE commands by typing the command on the EVE command line or by using the predefined keys.
- Move the cursor using the arrow keys or by entering EVE cursor movement commands on the command line.

Creating and Modifying a Text File

- EVE has two text entry modes: insert and overstrike.
- Use the REPLACE command to replace one character string with another.

Requesting Help with EVE

- Enter the EVE HELP command to request information on EVE topics.
- Press the **[Help]** key to display a diagram of the keypad.

Using Some Advanced Features of EVE

- Use the INCLUDE FILE command to include the contents of another file in the file being edited.
- Use the EVE DCL command to issue a DCL command from inside EVE.
- Use the SET KEYPAD command to set the keypad to emulate another editor.

Saving an Edited File

- The QUIT command ends an EVE editing session without making any changes to the file.
- The EXIT command ends an EVE editing session and saves a new version of the edited file.
- If you are editing a file when a system crashes or if you accidentally press **[Ctrl/Y]** while editing, you may recover your edits.

Commands

Performing Basic EVE Operations

EDIT [filename.filetype]

DCL command to invoke EVE.

Creating and Modifying a Text File

REPLACE [old-string] [new-string]

EVE command to replace one character string with another.

Requesting Help with EVE

HELP [topic]

EVE command to request a listing of EVE commands.

[Help]

Key that displays the EVE keypad diagram.

Using Some Advanced Features of EVE

INCLUDE FILE filename.type

EVE command to include another file into the current file. <ENDMARK>

DCL command

EVE command to issue a DCL command from the EVE command line.

SPAWN command

EVE command to spawn a subprocess to invoke an OpenVMS utility.

SET KEYPAD keypad-type

EVE command to set the application keypad to a specific editor.

Saving an Edited File

SAVE FILE

EVE command to write a new version of the edited file, using the current file name.

SAVE FILE AS

EVE command to write the current file as a new file using the specified file name.

QUIT

EVE command to end the editing session without writing a new version of the file.

EXIT

EVE command to exit the editing session and write a new version of the edited file.

EDIT/RECOVER original-file-spec

DCL command to recover an editing session.

Manipulating Files

Module 5

About This Module

Introduction

During the course of a terminal session, it is often necessary for the user to manipulate files. There are several common DCL commands that every user should be able to use.

This module discusses the use of the COPY, PRINT, RENAME, DELETE, and PURGE commands.

Objectives

To manipulate files, a user should be able to:

- Make copies of files.
- Display the contents of files.
- Print files.
- Rename files.
- Remove files from disk.

Resources

- OpenVMS DCL Dictionary
- OpenVMS User's Manual



Note

OpenVMS Version 7.2 introduces Extended File Specifications (EFS), implemented through Files-11 ODS-5, as well as RMS, DCL, RMS, and utility enhancements.

ODS-2 (the OpenVMS file system) provides:

- Limited number of characters in file names (39 for name, 39 for type)
- Limited character set for name (Alphanumeric, "\$", "-", "_")
- No support for lower/mixed case file names
- 8 level directory support, without using rooted logicals
- Advanced Server (the product formerly known as Pathworks) addressed some of these issues.

With OpenVMS 7.2, EFS supports (if implemented on the system):

- ISO-Latin (1 byte) and Unicode (2 byte) characters for file names
- Longer file names (236 characters ISO-Latin/118 Unicode including ".")
- More levels of directory (up to 255, this feature is also available under ODS-2 volumes)
- Non-standard characters in file names (all but C0 control (0-1F₁₆) / \ | : < > * ? ")

Case sensitive creation/Case insensitive lookups

Making Copies of Files

Overview

You may want to move a copy of a file into another directory, another disk, or even to another system.

One of the easiest and most common methods of moving files from one location to another is to use the `COPY` command.

This section discusses:

- `COPY` command
- Modifying the `COPY` command
- Common errors in copying files

COPY Command

Copying Files

To copy one or more existing files to a new file, use the COPY command.

The COPY command can do the following:

- Copy an input file to an output file
- Concatenate two or more input files into a single output file
- Copy a group of input files to a group of output files

Format of the COPY Command

COPY input-filespec[,...] output-filespec

- The input files remain unchanged.
- The output file contains the same text or data as the input file.

If you omit the input or output file specifications, you will be prompted for them.

```
$ COPY
 _From: INPUT.TXT
 _To: OUTPUT.FILE
```

Requirement for the COPY Command

To copy a file, you must have read access to the file.

Sample COPY Operations and Commands

If you want to copy...	Use a command like this:
One file to another in the same directory	\$ COPY TEMPLATE.REPORT APRIL.REPORT
One file to another in a different directory	\$ COPY MONTHLY.REPORT [.REPORTS]MARCH.STATUS
Two or more input files into a single output file (concatenation)	\$ COPY DEPT1.DIS,DEPT2.DIS MASTER.DIS
A group of input files to a group of output files in a different directory, using wildcards	\$ COPY *.TXT;* [SMITH.NEWBOOK]*
A file from a specified device and directory to your current default device and directory, keeping the same file name and file type	\$ COPY DJAO:[SOURCE]SAMPLE.TXT *
A file from a specified node, device, and directory to your current default device and directory, keeping the same file name and file type	\$ COPY NEAT::DJA0:[SOURCE] SAMPLE.TXT *

Modifying the COPY Command

Qualifiers

You can add qualifiers to the COPY command to modify its action. Qualifiers may be combined to further tailor a copy operation.

If you want the system to...	Use this qualifier:
Request confirmation before each copy operation	/CONFIRM
Display the file specification of each file copied	/LOG
Set a specific protection on the output file	/PROTECTION

Requesting Confirmation of the COPY Operation

Use the /CONFIRM qualifier to have the COPY command prompt the user for confirmation before each copy operation.

Format:

COPY/CONFIRM filename.filetype filename.filetype

You can use any combination of uppercase and lowercase letters for word responses, and they may be abbreviated.

- Abbreviations must be unique.
- Affirmative answers are YES, TRUE, and 1.
- Negative answers include: NO, FALSE, 0, and **[Return]** .
- Enter QUIT or **[Ctrl/Z]** to stop processing at that point.
- When you respond by entering ALL, the command continues to process but no further prompts are given.
- If you type a response other than one of those in the list, DCL issues an error message and redisplay the prompt.

Example: Copy Operation Using /CONFIRM

The following example shows a copy operation using the /CONFIRM qualifier.

```
❶ $ COPY/CONFIRM *.DAT [.DATA]*.*
❷ COPY WORK3:[USER1]HISTORY.DAT;1 to WORK3:[USER1.DATA]HISTORY.DAT;1 ? [N]: N
❸ COPY WORK3:[USER1]INVENTORY.DAT;1 to WORK3:[USER1.DATA]INVENTORY.DAT;1 ? [N]:
Y
COPY WORK3:[USER1]PROJECT.DAT;1 to WORK3:[USER1.DATA]PROJECT.DAT;1 ? [N]: Y
$
```

Example Notes: Copy Operation Using /CONFIRM

- ❶ Issue the command to copy all files having a file type of DAT to the DATA subdirectory. Specify the /CONFIRM qualifier.
- ❷ This is the COPY confirmation message. Type N or press **[Return]** to specify that this file is not to be copied.
- ❸ Type Y to request that this file be copied.

Displaying a Log of the COPY Operation

Use the /LOG qualifier to display the file specifications of each file copied.

Format:

COPY/LOG filename.filetype filename.filetype

The following information will be displayed after the copy operation:

- The file specifications of the input and output files
- The number of blocks copied
- The total number of new files created

Example: Copy Operation Using /LOG

The following command copies all files having a file type of REP to the REPORTS subdirectory. A log of the operation is requested.

```
$ COPY/LOG *.REP [.REPORTS]*.*  
❶ %COPY-S-COPIED, WORK3:[USER1]06DEC.REP;1 copied to  
   WORK3:[USER1.REPORTS]06DEC.REP;1  
   (7 blocks)  
   %COPY-S-COPIED, WORK3:[USER1]13DEC.REP;1 copied to  
   WORK3:[USER1.REPORTS]13DEC.REP;1  
   (10 blocks)  
   %COPY-S-COPIED, WORK3:[USER1]20DEC.REP;1 copied to  
   WORK3:[USER1.REPORTS]20DEC.REP;1  
   (9 blocks)  
❷ %COPY-S-NEWFILES, 3 files created
```

Example Notes: Copy Operation Using /LOG

- ❶ This is the log message displayed when a file is copied.
- ❷ This is the summary log message displayed when all files matching the specification have been processed.

Setting a New Protection Code on a Copied File

Use the /PROTECTION qualifier to set a new UIC protection code on a copied file.

Format:

COPY/PROTECTION=(prot-code) input-spec output-spec

- The default protection is that of the existing output file.
- If no output file exists, the current default protection applies.

Example: Copy Operation Using /PROTECTION

The following command copies the file MY.DAT to a new file called YOUR.DAT, specifying a new protection code

```
$ COPY/PROTECTION=(W:RE,S:RWED,O:RWED,G:RWED) MY.DAT YOUR.DAT
```

Common Errors in Copying Files

Errors in Copying Files

Errors in copying files usually involve opening the input or the output file. Check the text of the error message to determine which kind of error has occurred.

Example: Error in Input File

The following is an example of an error in the input specification for the COPY command.

```
$ COPY SAMPLE.COM [.COMMANDS]*.*
%COPY-E-OPENIN, error opening WORK3:[USER1]SAMPLE.COM; as input
-RMS-E-FNF, file not found
$
```

- In this case, the input file was not found.
- The input file could also be protected against you.

Example: Error in Output File

The following is an example of an error in the output specification for the COPY command. The output directory is protected against you.

```
$ COPY TEST.LOG [USER2]*.*
%COPY-E-OPENOUT, error opening WORK3:[USER2]TEST.LOG;1 as output
-RMS-E-PRV, insufficient privilege or file protection violation
%COPY-W-NOTCOPIED, WORK3:[USER1]TEST.LOG;1 not copied
$
```

Displaying the Contents of Files

Overview

One way to display the contents of a file is to edit the file. An editor will display the file and allow you to scroll forward and backward through the document.

Sometimes you do not need the ability to move forward and backward through a document, and it is much quicker to use the TYPE command to display the contents of a file at your terminal screen.

This section discusses:

- TYPE command
- Requirements and restrictions for the TYPE command
- Different ways to use the TYPE command

TYPE Command

Displaying the Contents of a File

Use the `TYPE` command to display the contents of a file at your terminal.

Format of the TYPE Command

Format:

TYPE file-spec

Specifying a file specification:

- The parameter consists of one or more file specifications separated by commas.
- You do not always have to specify every part of a file specification, because `TYPE` will supply default values.

Defaults for the TYPE Command

The file specification defaults used by the `TYPE` command are shown below.

Format of the TYPE Command

If left blank...	Will default to:
Device	Current default device
Directory	Current default directory
File name	No default
File type	LIS
Version	Highest version number

Example: Using the TYPE Command

In the following example, the `TYPE` command is used to display the contents of a file called `DEPT2.DIS`.

```
$ TYPE DEPT2.DIS
NEAT: :MARSH
TCHR: :MOREAU
TCHR: :PICCADACI
TIDY: :REGNELL
TIDY: :ROUNDS
$
```

Requirements and Restrictions for the TYPE Command

Requirement for the TYPE Command

To display the contents of a file, you must have read access to the file.

What Kinds of Files Can be Displayed

Files that are made up of ASCII characters, (A through Z, a through z, 0 through 9, and special characters available on the keyboard) can be typed. These are sometimes called text files.

Files that Should Not be Typed

Files that contain non-ASCII characters such as page breaks, form feeds, or other control sequences that send instructions to the operating system, should not be typed. When a user attempts to type these files, the operating system will try to carry out the instructions issued by the control characters. This can sometimes lock your terminal screen.

- Executable images
 - Easily identifiable by the .EXE file type
 - Example: PROGRAM.EXE
- Some data files
 - Usually have .DAT file type
 - Not all data files contain non-ASCII characters. Check if you are unsure.
 - Example: FILE.DAT
- Directory files
 - Easily identifiable by the .DIR file type
 - Example: MEMO.DIR

What to Do if You Display a Non-ASCII File

If you accidentally type a non-ASCII file and your terminal becomes locked:

- Press **[Set-Up]** on your terminal to access the setup menu.
- Select **[Clear Com]** and **[Reset Terminal]** to reset the terminal.
- Press **[Set-Up]** again to close the setup menu.

Different Ways to Use the TYPE Command

Displaying Multiple Files

Use a list of files or wildcards to display the contents of more than one file.

List of files:

- If more than one file is listed in the TYPE command, the files are displayed in the order specified.
- The following command will display the files in the order in which they are listed:

```
$ TYPE FILE03.TXT,FILE02.TXT,FILE01.TXT
```

Using wildcards:

- If wildcard characters are used, the files are displayed in alphabetical order.
- The command `$ TYPE FILE*.TXT` will display the files in the following order:

```
FILE01.TXT  
FILE02.TXT  
FILE03.TXT
```

Displaying One Screen at a Time

Use the TYPE/PAGE command to cause the contents of a file to be displayed one screen at a time.

Format:

TYPE/PAGE file-spec

When the screen is full, the system will display the following message at the bottom of the screen:

[Press RETURN to continue]

To abort the display, type any other character followed by **[Return]** .

If more than one file has been requested, you can cancel the display of the current file and continue with the next file by pressing Ctrl/Z.

The default is TYPE/NOPAGE.

Displaying a File in a Different Location

If a file and the directory that contains it are not protected against you, you can display the contents of a file in a different device, directory, or even on another system.

In a different device and directory:

```
$ TYPE $1$DUA1:[SMITH]COSTS.TXT
```

On a different system:

```
$ TYPE NEAT::$1$DUA1:[SMITH]COSTS.TXT
```

Printing Files

Overview

Although there is movement in the direction of the paperless office, it is still important to be able to have a physical copy of some documents. The OpenVMS operating system allows users to print files by sending a copy of a file to a printer (called an output device).

Many users send copies of files to printers all the time. There are many different kinds of printers, some with special paper, some with forms. A computer system uses a system of queues, or lines of requests, to keep track of which jobs should be printing where and on what printer.

This section discusses:

- PRINT command
- Modifying the action of the PRINT command
- Displaying information about print jobs
- Controlling print jobs
- Common errors in manipulating print jobs

PRINT Command

Printing Files

Use the PRINT command to send the contents of files to an output device such as a printer.

Format of the PRINT Command

The PRINT command has the following format:

PRINT filespec,[filespec...]

- You can supply a single file or a list of files, separated by commas, as the parameter to the PRINT command.
- The PRINT command uses a default file type of LIS if you do not specify another type.

Requirement for the PRINT Command

To print a file, you must have read access to the file. This means that the file must be on a disk and in a directory that you can read. The file also must be available for you to read.

Entry Number

When you issue the PRINT command, the OpenVMS operating system assigns an entry number to your print job. This number is used to identify the job and can be referenced by the user or system manager to control or manipulate the job in the queue.

**Note**

A job is a collection of subprocesses running under a detached process.

This number is displayed in an informational message when the PRINT command is issued. Entry numbers indicate the order in which jobs are queued.

In the following example, the system assigns entry number 776 to a print request.

```
$ PRINT TEST.DATA
Job TEST (queue SYS$PRINT, entry 776) started on LPA0
$
```

Modifying the Action of the PRINT Command

Qualifiers for the PRINT Command

You can use a variety of qualifiers to modify the action of the PRINT command. Qualifiers may be combined to further tailor a print operation. Some of the most common qualifiers are listed below.

If you want to...	Use this qualifier:
Specify a particular queue	/QUEUE
Specify the number of copies	/COPIES
Request notification when the print job completes	/NOTIFY

Specifying a Queue

There is often more than one printer connected to a system. Some printers may be designated to handle large jobs; some to handle small, quick jobs; some to handle special forms. These queues will be given names by the system manager and can be referenced in the print request to send a particular job to a specific queue.

Specify a particular queue to print a job by using the /QUEUE qualifier:

```
$ PRINT/QUEUE=LETTER_QUALITY MONTHLY_REPORT.TXT
```

If you do not specify a queue, your file will be sent to the default system queue, SYSS\$PRINT.

Specifying the Number of Copies Wanted

Use the `/COPIES=n` qualifier to specify the number of copies you want.

- The number of copies is represented by *n* and can be a number from 1 to 255. The default is 1.
- The following command prints two copies of the file `MEMO.TXT`:

```
$ PRINT/COPIES=2 MEMO.TXT
```

The position of the `/COPIES` qualifier in the command determines its effect.

- When it is attached to the `PRINT` command, the `/COPIES` qualifier affects all files in the print job.

The following command prints two copies each of `Q1.TXT` and `Q2.TXT`:

```
$ PRINT/COPIES=2 Q1.TXT,Q2.TXT
```

- When it is attached to a file specification, the `/COPIES` qualifier only affects that one file.

The following command prints two copies of `Q1.TXT` and one copy of `Q2.TXT`:

```
$ PRINT Q1.TXT/COPIES=2,Q2.TXT
```

Asking the System to Notify on Completion

Use the `/NOTIFY` qualifier to ask the system to broadcast a message at your terminal when a print job has completed.

Example:

```
$ PRINT/NOTIFY LOGIN.COM
Job LOGIN (queue SYS$PRINT, entry 799) started on LPS40$SCDTST
$
%LPS-I-JOBSTART, Job 799 Start
$
Job LOGIN (queue SYS$PRINT, entry 799) completed
$
```

Displaying Information About Print Jobs

Monitoring Print Jobs

By monitoring the progress of your print requests, you can intervene to adjust your request to better fit your immediate needs and the needs of the system.

Use the `SHOW ENTRY` command or the `SHOW QUEUE` command to display the status of jobs in a print queue.

SHOW ENTRY Command

Use the `SHOW ENTRY` command to display information about your print jobs or about specific job entries.

Format:

SHOW ENTRY [entry-number,...] [job-name]

- The display shows each entry's current status as well as its attributes, including entry number, job name, owner, job status, and queue name.
- Add the `/FULL` qualifier to display more detailed information.
- If you do not specify an entry number or job name, only information about your queue entries will be displayed.

Example: Listing Queue Entries

The following example shows the output from the `SHOW ENTRY` command.

```
$ SHOW ENTRY
Entry Jobname      Username   Blocks Status
-----
 970 MONTHLY      USER1      25 Printing
    On busy printer queue LTA1
 973 EX_COM_PROC  USER1     175 Pending
    On Generic printer queue POST
$
```

Example: Displaying a Specific Queue Entry

The following example uses the `SHOW ENTRY` command to display information about a specific queue entry.

```
$ SHOW ENTRY 970
Entry Jobname      Username   Blocks Status
-----
 970 MONTHLY      USER1      25 Printing
    On busy printer queue LTA1
$
```

Example: Displaying Detailed Queue Information

The following example uses the SHOW ENTRY/FULL command to display detailed information about all of your queue entries.

```
$ SHOW ENTRY/FULL
Entry Jobname      Username    Blocks Status
-----
970 MONTHLY        USER1       25 Printing
    On busy printer queue LTA1
Submitted 14-AUG-1995 12:35 /FORM=DEFAULT /PRIORITY=100
    File: _$1$DUA7:[USER1]MONTHLY.PS;1 (printing)

973 EX_COM_PROC    USER1       175 Pending (check execution queues)
    On Generic printer queue POST
Submitted 14-AUG-1995 12:36 /FORM=DEFAULT /NOTIFY /PRIORITY=100
    File: _$1$DUA7:[USER1.BUILD]EX_COM_PROC.PS;3

$
```

SHOW QUEUE Command

The SHOW QUEUE command can display information about queues as well as about jobs in a queue.

Format:

SHOW QUEUE [queue-name]

- If you do not specify a queue name, all queues will be displayed.
- Enter SHOW QUEUE/DEVICES to list only the printer queues.

Example: Displaying Queue Information

The following example shows the output from the SHOW QUEUE command.

```
$ SHOW QUEUE
Printer queue LPS20$CRIER, on TIDY::CRIER, mounted form DEFAULT
Generic printer queue POSTB
Generic printer queue POSTO
Generic printer queue POST_20
  Entry Jobname      Username    Blocks Status
  -----
    976 CHAP_ONE     USER1      265 Pending
    979 CHAP_TWO     USER1      287 Pending
.
.
.
```

When you issue the SHOW QUEUE command without specifying a queue name or any qualifiers, only the jobs submitted by this user will be displayed, but all available queues will be displayed.

Controlling Print Jobs

Controlling Print Jobs

You can take a variety of actions to control your print jobs:

- Change the characteristics of a job
- Put your job on hold to print at a later time
- Release a holding job for immediate printing
- Remove a print request from the queue

Changing the Characteristics of Your Print Job

Use the SET ENTRY command to change certain characteristics of your print job.

Format:

SET ENTRY *entry-number* /*qualifier*

- The *entry-number* is your job number in the queue, displayed by the SHOW QUEUE or SHOW ENTRY command.
- Use the qualifier(s) to specify the characteristic(s) you wish to modify.

This command will not affect a job that is currently executing.

Job Characteristics You Can Change

Use SET ENTRY qualifiers to specify new characteristics for a print job. Some commonly used qualifiers are listed below.

If you want to...	Use this SET ENTRY qualifier:
Delay processing until a specified time or date	/AFTER=time
Specify or change the number of copies	/COPIES=n
Hold the print request until a later time or date	/HOLD
Have the system notify you when the print job completes	/NOTIFY
Release a holding job for processing	/RELEASE

Example: Changing Characteristics of a Print Job

The following example uses the SET ENTRY command to change the processing time and number of copies for a print job and request notification when the job has completed.

```
$ PRINT SETHOST.LOG
Job SETHOST (queue SYS$PRINT, entry 749) pending
$ SHOW ENTRY/FULL
Entry Jobname      Username    Blocks Status
-----
749 SETHOST      USER1      13 Pending
   On generic printer queue SYS$PRINT
   Submitted 14-AUG-1995 10:32 /FORM=DEFAULT /PRIORITY=100
   File: _$1$DUA1:[USER1]SETHOST.LOG;2
$ SET ENTRY/COPIES=2/NOTIFY/AFTER=11:00 749
$ SHOW ENTRY/FULL
Entry Jobname      Username    Blocks Status
-----
749 SETHOST      USER1      26 Holding until 14-AUG-1995 11:00
   On generic printer queue SYS$PRINT
   Submitted 14-AUG-1995 10:32 /FORM=DEFAULT /NOTIFY /PRIORITY=100
   File: _$1$DUA1:[USER1]SETHOST.LOG;2 /COPIES=2
$
.
.
.
Job SETHOST (queue SYS$PRINT, entry 749) completed
$
```

Example: Holding and Releasing a Job

The following example uses the SET ENTRY command to release a print job that is in a holding status.

```
$ PRINT/HOLD MARCH.REPORT
Job MARCH (queue SYS$PRINT, entry 112) holding
.
.
.
$ SET ENTRY/RELEASE 112
$
```

Removing Your Print Job from the Queue

Use the DELETE/ENTRY command to remove your job from a print queue. The job can be in progress or waiting in the queue.

Format:

DELETE/ENTRY((entry-number[,...]))

The *entry-number[,...]* parameter specifies the entry number (or a list of entry numbers) of jobs to be deleted.

- If you specify only one entry number, you can omit the parentheses.
- To find a job's entry number, enter the SHOW ENTRY or SHOW QUEUE command.

Requirements for Deleting a Job

The DELETE/ENTRY command requires one of the following:

- OPER (operator) privilege
- Execute (E) access to the queue
- Delete (D) access to the specified job(s)

All users have delete access to any job they have submitted. In addition, system managers and operators and any other system personnel that need to have control of a system will have the privilege to manipulate user jobs.

Example: Removing a Print Job from the Queue

In the following example, the user uses the DELETE/ENTRY command to delete a pending print job and one that is holding. Note the system message that is displayed when each type of job is deleted.

```
$ SHOW ENTRY
Entry Jobname      Username   Blocks Status
-----
25 CHAPTER8      USER1      976 Pending
    On generic printer queue SYS$PRINT

27 SETHOST       USER1      26 Holding until 14-AUG-1995 11:00
    On generic printer queue SYS$PRINT
$ DELETE/ENTRY=25
%DELETE-I-DELETED, entry 25 aborting or deleted
$ DELETE/ENTRY=27
Job SETHOST (queue SYS$PRINT, entry 27) terminated with error status
%JBC-F-JOBDELETE, job deleted before execution
%DELETE-I-DELETED, entry 27 aborting or deleted
$
```

Common Errors in Manipulating Print Jobs

Errors in Manipulating Print Jobs

When manipulating print jobs, common errors involve specifying a nonexistent:

- File
- Queue
- Job
- Entry

Error: Nonexistent File

In the following example, the user specifies a nonexistent file in the print request.

```
$ PRINT MISSING
%PRINT-E-OPENIN, error opening $1$DUA1:[USER1]MISSING.LIS; as input
-RMS-E-FNF, file not found
%PRINT-F-CREJOB, error creating job
-JBC-E-EMPTYJOB, no files specified in job request
$
```

What to do:

- Check your typing: did you misspell the file name?
- Check your default directory: is the file you want in the current directory?
- Note that the default file type for the PRINT command is LIS. Do you need to specify a different file type?

Error: Nonexistent Queue

In the following example, the user specifies a nonexistent queue in the print request.

```
$ PRINT/QUE=LPS40 LOGIN.COM
%PRINT-F-CREJOB, error creating job
-JBC-E-NOSUCHQUE, no such queue
$
```

What to do:

- Use the SHOW QUEUE/DEVICES command to display the names of print queues on your system.

Error: Nonexistent Job

In the following example, the user tries to delete a nonexistent job.

```
$ DELETE/ENTRY=2
%DELETE-W-SEARCHFAIL, error deleting 2
-JBC-E-NOSUCHENT, no such job
$
```

What to do:

- Use the SHOW ENTRY command to display your jobs, and check the job numbers.

It is a common error to specify the block size rather than the entry number. The entry number for the following job is 735, not 2.

```
$ SHOW ENTRY
Entry Jobname      Username      Blocks Status
-----
  735 LOGIN        USER1         2 Pending
On generic printer queue SYS$PRINT
$
```

Error: Nonexistent Entry

In the following example, the system message does not indicate a problem. The user has no jobs in the queue, probably because they have completed.

```
$ SHOW ENTRY
-JBC-E-NOSUCHENT, no such entry
$
```

Renaming Files

Overview

Sometimes it is necessary to rename a file. You might want to:

- Save a snapshot of a file at a particular time
- Save a specific version of a file
- Indicate differences in a routinely edited file

You could accomplish this task by first copying the file to a new file and then deleting the original file. The RENAME command will perform these operations.

This section discusses:

- RENAME command
- Modifying the RENAME command
- Common errors in renaming files

RENAME Command

Renaming a File

Use the RENAME command to change the name of a file or move a file to another directory on the same disk while deleting the file from its original location.

Format of the RENAME Command

The format of the RENAME command is:

RENAME input-filespec output-filespec

- The *output-filespec* may contain a different directory name than the *input-filespec*, but it must not specify a different disk.
- If you omit one or both of the file specifications, the RENAME command will prompt you for them:

```
$ RENAME
 _From: TEST.COM
 _To: PRACTICE.FILE
```

Requirements for the RENAME Command

To rename a file, you must have:

- Delete access to the file
- Write access to the directory containing the new file

Sample Rename Operations

If you want to...	Issue this command:
Change the highest version of the file FILE1.OBJ to FILE2.OBJ	\$ RENAME FILE1.OBJ FILE2
Rename all versions of all files with the file type TXT to file type OLD	\$ RENAME *.TXT;* *.OLD;*
Rename a file into another directory. (In effect, this moves the file from one directory to another.)	\$ RENAME TEST.COM [.COMMANDS]

Using Version Numbers with the RENAME Command

If there are multiple versions of a file, you must rename all versions or there will be copies of the older versions left on disk after the rename operation.

The RENAME command understands relative version numbers.

Highest version	;0
Second highest version	;-1
Tenth highest version	;-9
Lowest version	;-0

Example:

```
$ RENAME MONTHLY.REPORT;-1 .OLD
```

Modifying the RENAME Command

Qualifiers

You can add qualifiers to the RENAME command to modify its action. Qualifiers may be combined to further tailor a RENAME operation.

If you want the system to...	Use this qualifier:
Request confirmation before each rename operation	/CONFIRM
Display the file specification of each file renamed	/LOG

Requesting Confirmation of the Rename Operation

Add the /CONFIRM qualifier to the RENAME command to control whether a request is issued before each rename operation.

Format:

\$ RENAME/CONFIRM input-filespec output-filespec

You will be prompted for each file that matches the specification.

- Type N or press **[Return]** to prevent renaming.
- Type Y (or YES) to have the file renamed.

If you respond with ALL, you can rename all of the remaining files.

Example: Rename Operation Using /CONFIRM

The following command renames all files having a file name of SAMPLE to have a file name of DEMONSTRATION. The file types remain unchanged. The /CONFIRM qualifier is used to request confirmation of the operation.

```
$ RENAME/CONFIRM SAMPLE.* DEMONSTRATION.*
RENAME TIDY$DKA300:[USER1]SAMPLE.FILE;1 to TIDY$DKA300:[USER1]DEMONSTRATION.FILE; ?
[N]: N
RENAME TIDY$DKA300:[USER1]SAMPLE.INFO;1 to TIDY$DKA300:[USER1]DEMONSTRATION.INFO; ?
[N]: Y
$
```

Displaying a Log of a Rename Operation

Use the /LOG qualifier to cause the system to display the name of each file as it is renamed.

Format:

\$ RENAME/LOG input-filespec output-filespec

Example: Rename Operation Using /LOG

The following command renames all files having a file type of .DATA to have the same file name but the new file type .INFO. A log of the operation is requested.

```
$ RENAME/LOG *.DATA *.INFO
%RENAME-I-RENAMED, TIDY$DKA300:[USER1]PRELIMINARY.DATA;1 renamed
to TIDY$DKA300:[USER1]PRELIMINARY.INFO;1
%RENAME-I-RENAMED, TIDY$DKA300:[USER1]SAMPLE.DATA;1 renamed to
TIDY$DKA300:[USER1]SAMPLE.INFO;1
%RENAME-I-RENAMED, TIDY$DKA300:[USER1]TESTING.DATA;14 renamed to
TIDY$DKA300:[USER1]TESTING.INFO;1
$
```

Common Errors in Renaming Files

Errors in Renaming Files

Errors in renaming files usually involve opening the input file or writing to the output file. Check the text of the error message to determine which kind of error has occurred.

Example: Error in Input File

The following is an example of an error in the input specification for the RENAME command. The user does not have delete access to the input file.

```
$ RENAME TEST.COM [.COMMANDS]*.*
%RENAME-E-OPENIN, error opening TIDY$DKA300:[USER1]TEST1.COM;1 as input
-RMS-F-RMV, ACP remove function failed
-SYSTEM-F-NOPRIV, no privilege for attempted operation
$ DIRECTORY/PROTECTION TEST1.COM

Directory TIDY$DKA300:[USER1]

TEST1.COM;1      (RWE,RWE,RE, )

Total of 1 file.
$
```

Example: Error in Output File

The following is an example of an error in the output specification for the RENAME command. The user does not have write access to the output directory.

```
$ RENAME TEST2.COM [.DATA]
%RENAME-E-OPENIN, error opening TIDY$DKA300:[USER1]TEST2.COM;1 as input
-RMS-E-ENT, ACP enter function failed
-SYSTEM-F-NOPRIV, no privilege for attempted operation
$ DIRECTORY/PROTECTION DATA.DIR
Directory TIDY$DKA300:[USER1]
DATA.DIR;1      (RE,RE,RE,RE)
Total of 1 file.
$
```

Removing Files from Disk

Overview

There are several reasons why a user will need to be able to remove files from disk areas. Space is always a consideration. The OpenVMS operating system writes a new version of a file every time a user edits it, so versions can pile up very quickly and consume disk space.

It is also important to keep copies of files at a minimum to prevent confusion about which copy of a file is the one that a user wants to use.

This section discusses:

- DELETE command
- Modifying the DELETE command
- Common errors in deleting files
- PURGE command
- Modifying the PURGE command
- Common errors in purging files
- Erasing the contents of deleted files

DELETE Command

Deleting Files

Use the DELETE command to remove files and free disk space.

Note that there is no UNDELETE command. Once you delete a file, you usually cannot get it back.

Format of the DELETE Command

The DELETE command has the following format:

DELETE file-name.file-type;version_number

- You must specify the version number of the file you want to delete.
- If you do not specify the file to be deleted, you will be prompted for it.

```
$ DELETE
_File: TEST.COM;1
```

Specifying Version Numbers

If you want to delete...	Issue this command:
A specific version of a file	DELETE name.type;version \$ DELETE INVENTORY.LIS;3
The highest version of a file	DELETE name.type; \$ DELETE INVENTORY.LIS;
All versions of a file	DELETE name.type;* \$ DELETE INVENTORY.LIS;*



Note

When you delete a file, the area in which the file was stored is returned to the system for future use. The data that was stored in that location still exists in the system until new data is written over it. When you specify the /ERASE qualifier, the storage location is overwritten with a system specified pattern so that the data no longer exists.

Requirements for the DELETE Command

To delete a file, you must have:

- Delete access to the file
- Write access to the directory containing the file

Modifying the DELETE Command

Qualifiers

You can add qualifiers to the DELETE command to modify its action. Qualifiers may be combined to further tailor a delete operation.

If you want the system to...	Use this qualifier:
Request confirmation before each delete operation	/CONFIRM
Display the file specification of each file deleted	/LOG

Requesting Confirmation of a Delete Operation

The /CONFIRM qualifier is helpful when you use wildcards to specify the files to be deleted. This protects against accidentally deleting the wrong files.

You will be prompted for each file that matches the specification.

- Type N, 0, or F or press **[Return]** to prevent deletion.
- Type Y, 1, or T to have the file deleted.

If you respond with ALL, you can delete all of the remaining files.

```
$ DELETE/CONFIRM *.INFO;
DELETE TIDY$DKA300:[USER1]DEMONSTRATION.INFO;1 ? [N]: Y
DELETE TIDY$DKA300:[USER1]PRELIMINARY.INFO;1 ? [N]: N
DELETE TIDY$DKA300:[USER1]TESTING.INFO;1 ? [N]: Y
$
```

Requesting a Log of a Delete Operation

Deleting information is an operation that may require close monitoring to ensure that what the user wants to delete is being deleted. Use the /LOG qualifier to cause the system to display the name of each file as it is deleted.

```
$ DELETE/LOG *.DATA;
%DELETE-I-FILDEL, TIDY$DKA300:[USER1]SAMPLE.DATA;1 deleted (3 blocks)
%DELETE-I-FILDEL, TIDY$DKA300:[USER1]TESTING.DATA;11 deleted (3 blocks)
%DELETE-I-FILDEL, TIDY$DKA300:[USER1]TRIAL.DATA;1 deleted (3 blocks)
%DELETE-I-TOTAL, 3 files deleted (9 blocks)
$
```

Common Errors in Deleting Files

Errors in Deleting Files

Errors in deleting files usually involve failing to specify a version number or not having delete access to the file. Check the text of the error message to determine the kind of error that has occurred.

No Version Number Specified

The following error occurred because the user did not specify a version number for the delete operation.

```
$ DELETE MEMO.TXT
%DELETE-E-DELVER, explicit version number or wild card required
$
```

Any of the following commands would be correct:

\$ DELETE MEMO.TXT;	Delete the highest version
\$ DELETE MEMO.TXT;3	Delete a specific version
\$ DELETE MEMO.TXT;*	Delete all versions

Trying to Delete a Protected File

The following error occurred because the user does not have delete access to the specified file.

```
$ DELETE COMMANDS.DIR;1
%DELETE-W-FILNOTDEL, error deleting TIDY$DKA300:[USER1]COMMANDS.DIR;1
-RMS-E-PRV, insufficient privilege or file protection violation
$
```

What to do:

- If you do not own the file, arrange to be given sufficient privilege to delete the file, request that the protection on the file be changed, or ask the system manager or file owner to perform the deletion.
- If you are the owner of the file, change the protection on the file to give yourself D access and reissue the DELETE command.

PURGE Command

Purging Files

The OpenVMS operating system creates a new version of a file every time you modify that file in any way.

Purging old versions of a file helps conserve disk space. Use the PURGE command to remove all but the highest version of a file from the disk.

Format of the PURGE Command

The PURGE command has the following format

PURGE [file-specification]

- Do not include a version number with the file specification.
- If you omit the file specification, all files in the current default directory will be purged.

Requirement for the PURGE Command

To purge a file, you must have:

- Delete access to the file
- Write access to the directory containing the file

Modifying the PURGE Command

Qualifiers

You can add qualifiers to the PURGE command to modify its action. Qualifiers may be combined to further tailor a purge operation.

If you want the system to...	Use this qualifier:
Request confirmation before each operation	/CONFIRM
Display the file specification of each file deleted	/LOG
Specify the number of versions of the file to be retained	/KEEP

Requesting Confirmation of the Purge Operation

The /CONFIRM qualifier is helpful when you use wildcards to specify the files to be purged. This protects against accidentally deleting the wrong files.

Format:

\$ PURGE/CONFIRM filespec

You will be prompted for each file that matches the specification.

- Type N, 0, or F, or press **[Return]** to prevent purging.
- Type Y, 1, or T to have the file purged.

If you respond with ALL, you can purge all of the remaining files.

Example: Purge Operation Using /CONFIRM

The following command purges all files with a file type of .DATA, and requests confirmation of the operation.

```
$ PURGE/CONFIRM *.DATA
DELETE TIDY$DKA300:[USER1]SAMPLE.DATA;2 ? [N]: N
DELETE TIDY$DKA300:[USER1]SAMPLE.DATA;1 ? [N]: Y
DELETE TIDY$DKA300:[USER1]TRIAL.DATA;1 ? [N]: Y
$
```

Displaying a Log of the Purge Operation

Use the /LOG qualifier to cause the system to display the name of each file as it is deleted.

Format:

\$ PURGE/LOG filespec

Example: Purge Operation Using /LOG

The following command purges all files having a file type of .INFO, and requests a log of the operation.

```
$ PURGE/LOG *.INFO
%PURGE-I-FILPURG, TIDY$DKA300:[USER1]SAMPLE.INFO;1 deleted (3 blocks)
%PURGE-I-FILPURG, TIDY$DKA300:[USER1]TEST.INFO;5 deleted (3 blocks)
%PURGE-I-FILPURG, TIDY$DKA300:[USER1]TEST.INFO;4 deleted (3 blocks)
%PURGE-I-FILPURG, TIDY$DKA300:[USER1]TEST.INFO;3 deleted (3 blocks)
%PURGE-I-FILPURG, TIDY$DKA300:[USER1]TEST.INFO;2 deleted (3 blocks)
%PURGE-I-FILPURG, TIDY$DKA300:[USER1]TEST.INFO;1 deleted (3 blocks)
%PURGE-I-TOTAL, 6 files deleted (18 blocks)
$
```

Keeping More than One Version of a File

Use the /KEEP qualifier to retain more than one version of a file.

Format:

PURGE/KEEP=*n* [file-spec]

- The parameter *n* indicates the number of versions you wish to keep.
- The default is to keep only one version of the file, the highest-numbered version.

Example: Purge Operation Using /KEEP

The following command purges a file called KEEP.TXT, keeping the two highest versions.

```
$ DIRECTORY KEEP.TXT
Directory TIDY$DKA300:[USER1]
KEEP.TXT;4      KEEP.TXT;3      KEEP.TXT;2      KEEP.TXT;1
Total of 4 files.
$ PURGE/KEEP=2 KEEP.TXT
$ DIRECTORY KEEP.TXT
Directory TIDY$DKA300:[USER1]
KEEP.TXT;4      KEEP.TXT;3
Total of 2 files.
$
```

Example: Combining PURGE Command Qualifiers

You can tailor the PURGE command by combining qualifiers. The following example combines the /LOG and /KEEP qualifiers.

```
$ DIRECTORY/COLUMNS=1 MAILING_LIST.TXT
Directory TIDY$DKA300:[USER1]
MAILING_LIST.TXT;27

.
.
.
MAILING_LIST.TXT;12

Total of 16 files.
$ PURGE/LOG/KEEP=10 MAILING_LIST.TXT
%PURGE-I-FILPURG, TIDY$DKA300:[USER1]MAILING_LIST.TXT;17 deleted (909 blocks)

.
.
.
%PURGE-I-FILPURG, TIDY$DKA300:[USER1]MAILING_LIST.TXT;12 deleted (2739 blocks)
%PURGE-I-TOTAL, 6 files deleted (13383 blocks)
$
```

Common Errors in Purging Files

Errors in Purging Files

Errors in purging files usually involve specifying a version number or not having delete access to the file. Check the text of the error message to determine the kind of error that has occurred.

Example: Specifying a Version Number When Purging Files

The following error occurred because the user specified a version number with the PURGE command.

```
$ PURGE TEST.*;  
%PURGE-E-PURGEVER, version numbers not permitted  
$
```

Do not include a semicolon (;) in the PURGE command.

```
$ PURGE TEST.*
```

Example: Trying to Purge a Protected File

The following error occurred because the user does not have delete access to the specified file.

```
$ DIRECTORY/PROTECTION PROT.FILE  
Directory TIDY$DKA300:[USER1]  
PROT.FILE;3      (RE,RE,RE,)  
PROT.FILE;2      (RE,RE,RE,)  
PROT.FILE;1      (RE,RE,RE,)  
  
Total of 3 files.  
$ PURGE PROT.FILE  
%PURGE-W-FILNOTPUR, error deleting TIDY$DKA300:[USER1]PROT.FILE;2  
-RMS-E-PRV, insufficient privilege or file protection violation  
%PURGE-W-FILNOTPUR, error deleting TIDY$DKA300:[USER1]PROT.FILE;1  
-RMS-E-PRV, insufficient privilege or file protection violation  
$
```

Example: Trying to Purge a File in a Protected Directory

The following error occurred because the user does not have write access to the directory containing the specified file. Delete access to the file itself is not sufficient.

```
$ DIRECTORY/PROTECTION TEST.DIR
```

```
Directory TIDY$DKA300:[USER1]
```

```
TEST.DIR;1      (RE,RE,RE,RE)
```

```
Total of 1 file.
```

```
$ DIRECTORY/PROTECTION [.TEST]
```

```
Directory TIDY$DKA300:[USER1.TEST]
```

```
ABC.DATA;3      (RWED,RWED,RE,)
```

```
ABC.DATA;2      (RWED,RWED,RE,)
```

```
ABC.DATA;1      (RWED,RWED,RE,)
```

```
Total of 3 files.
```

```
$ PURGE [.TEST]
```

```
%PURGE-W-FILNOTPUR, error deleting TIDY$DKA300:[USER1.TEST]ABC.DATA;2
```

```
-RMS-E-PRV, insufficient privilege or file protection violation
```

```
%PURGE-W-FILNOTPUR, error deleting TIDY$DKA300:[USER1.TEST]ABC.DATA;1
```

```
-RMS-E-PRV, insufficient privilege or file protection violation
```

```
$
```

Erasing the Contents of Deleted Files

Contents of Deleted Files

When you delete a file, the area in which the file was stored is returned to the system for future use. The information that was stored in that location still exists in the area until new information is written over it. A person with advanced knowledge may be able to access the data in a file that has been deleted, but not overwritten.

Erasing Files

To prevent unauthorized access to a deleted confidential file, use the /ERASE qualifier when issuing the DELETE or PURGE commands. This causes the storage location to be overwritten with a system specified pattern so that the previous information no longer exists.

Syntax:

DELETE file-specification/ERASE

PURGE file-specification/ERASE

Summary

Concepts

Making Copies of Files

- To copy one or more existing files to a new file, use the COPY command.
- To copy a file, you must have read access to the file.
- You can use a variety of qualifiers to modify the action of the COPY command.
- Errors in copying files usually involve opening the input or the output file.

Displaying the Contents of Files

- Use the TYPE command to display the contents of a file at your terminal.
- If you do not supply all parts of a file specification, the TYPE command assumes certain default values.
- To display the contents of a file, you must have read access to the file.
- Use lists of files or wildcards to display multiple files.
- Use the TYPE/PAGE command to display a file one screen at a time.

Printing Files

- To send the contents of a file(s) to an output device such as a printer, use the PRINT command.
- To print a file, you must have read access to the file.
- When you issue the PRINT command, the OpenVMS operating system assigns an entry number to your print job.
- You can use a variety of qualifiers to modify the action of the PRINT command. Qualifiers may be combined to further tailor a print operation.
- To display the status of jobs in a print queue, use the SHOW ENTRY command or the SHOW QUEUE command.
- To change certain characteristics of your print job, use the SET ENTRY command. This command will not affect a job that is currently printing.
- To remove your job from a print queue, use the DELETE/ENTRY command. The job can be in progress or waiting in the queue.
- Common errors in manipulating print jobs involve specifying a nonexistent file, queue, job, or entry.

Renaming Files

- To change the name of a file or move a file to a different directory on the same disk, use the RENAME command.
- To rename a file, you must have delete access to the file and write access to the directory containing the file.
- You can use qualifiers to modify the action of the RENAME command.
- Errors in renaming files usually involve opening the input file or writing to the output file.

Removing Files from Disk

- To remove specified versions of a file from a disk, use the DELETE command. You must specify the version number in some manner.
- To delete a file, you must have delete access to the file and write access to the directory containing the file.
- You can add qualifiers to the DELETE command to modify its action.
- Qualifiers may be combined to further tailor a delete operation.
- Errors in deleting files usually involve failing to specify a version number or not having delete access to the file.
- To remove all but the highest version of a file from a disk, use the PURGE command. You must not specify a version number.
- To purge a file, you must have delete access to the file and write access to the directory containing the file.
- You can add qualifiers to the PURGE command to modify its action.
- Qualifiers may be combined to further tailor a purge operation.
- Errors in purging files usually involve specifying a version number or not having delete access to the file.
- For further protection, delete sensitive files with the /ERASE qualifier.

Commands

Making Copies of Files

COPY

Copy one or more existing files to a new file.

Displaying the Contents of Files

TYPE

Display the contents of a file.

Printing Files

PRINT

Send the contents of a file to an output device.

SHOW ENTRY

Display status information about jobs in a print queue.

SHOW QUEUE

Display status information about print queues.

SET ENTRY

Change characteristics of your print job.

DELETE/ENTRY

Remove your print job from a print queue.

Renaming Files

RENAME

Change the name of a file or move the file to another directory on the same disk.

Removing Files from Disk

DELETE

Remove files from a disk.

PURGE

Remove previous versions of files from a disk.

Communicating with Other Users

Module 6

About This Module

Introduction

One of the most useful features of an OpenVMS system is its built-in electronic mail capability. The OpenVMS Mail utility allows you to exchange messages with users on your system, and with users on other nodes in your network. In addition, you can:

- Reply to, forward, print, or file messages that you receive
- Organize your messages by grouping them together in folders
- Use distribution lists to send messages to any number of users
- Extract a mail message into a file

This module shows how to use the Mail utility commands that allow you to perform these and other operations.

Objectives

To communicate with other users, a user should be able to use Mail utility commands to:

- Send and receive messages using the Mail utility.

Resources

- *OpenVMS User's Manual*

Using the Mail Utility

Overview

You can use the OpenVMS Mail utility (MAIL) to send messages to other users on your system or on any other computer that is connected to your system by means of DECnet for OpenVMS systems. You can also read, file, forward, delete, print, and reply to messages that other users send to you.

This section discusses:

- Getting started with the Mail utility
- Displaying information about current messages
- Reading messages
- Sending messages
- Deleting messages
- Additional Mail features

Getting Started with the Mail Utility

Overview

To get started using the Mail utility, you need to know:

- How to invoke the Mail utility
- How mail folders are used
- How to exit from the Mail utility

Invoking the Mail Utility

Use the MAIL command to invoke the OpenVMS Mail utility. The Mail prompt will be displayed.

```
$ MAIL
```

```
MAIL>
```

Mail Folders

All mail messages are organized into **folders**.

- A folder is much like a drawer in a file cabinet: an organized place for storing related messages.
- Some folders are created by the Mail utility.
 - The NEWMAIL folder contains messages that you have not read.
 - ◆ After messages in the NEWMAIL folder have been read, they are automatically moved into the MAIL folder.
 - ◆ The NEWMAIL folder automatically disappears after you have read all of your new messages.
 - The MAIL folder contains messages that you have read.
 - The WASTEBASKET folder contains any messages that you have deleted.
- The user can create additional folders.

By default, your mail file (MAIL.MAI) contains a folder named MAIL. This folder contains messages you have already read.

New messages automatically enter a folder named NEWMAIL.

When you delete a message, it is placed in the WASTEBASKET folder, where it remains until you exit from the utility. You can retrieve a message from this folder if you have not exited after deleting it.

**Note**

The wastebasket will be automatically emptied only if the SET AUTO_PURGE command has been entered. Note also that if AUTO_PURGE is set, the WASTEBASKET folder is also emptied when a user switches files using the SET FILE command.

Figure: Default Mail Folders



The following figure shows the three folders the Mail utility creates for you.

Example: Identifying Your Current Folder

The name of the folder you are currently using is displayed at the top right corner of your screen when you enter the READ or DIRECTORY command.

In the following example, MAIL is the current folder.

```
MAIL> DIRECTORY
```

			MAIL
#	From	Date	Subject
1	NEAT::USER2	12-JAN-1999	Team meeting
2	TIDY::USER1	14-JAN-1999	RE: Project plan
3	TIDY::SYSTEM	22-JAN-1999	Access to VMS\$COM

```
MAIL>
```

Exiting from the Mail Utility

There are three ways to exit from the Mail utility.

To exit from the Mail utility...	Enter:
And empty the WASTEBASKET folder	EXIT
	(or)
	[Ctrl/Z]
Without emptying the WASTEBASKET folder	QUIT

Displaying Information About Messages

Listing Mail Messages

The Mail DIRECTORY command displays a numbered list of your mail messages.

MAIL> DIRECTORY

Example: Listing Mail Messages

```
MAIL> DIRECTORY
1                                     2      MAIL
3 4                                5      6
# From                          Date      Subject

1 NEAT::USER2                   12-JAN-1999 Tomorrow's meeting
2 TIDY::USER1                    14-JAN-1999 RE: Tomorrow's meeting
.
.
.
14 NEAT::USER2                   18-JAN-1999 Minutes of team meeting

MAIL>
```

Example Notes: Listing Mail Messages

- ❶ Request a list of your mail messages. (There are 14 messages.)
- ❷ This is a listing of the main MAIL folder.
- ❸ Each message is numbered, allowing you to specify the message you wish to manipulate.
- ❹ The node and user name of the message sender.
- ❺ The date the message was received.
- ❻ The sender's statement of what the message is about. If the subject is left blank, or is filled in with For Your Information (FYI), that is not helpful to the recipient who is looking for information about the contents of the message.



Note

To view a list of mail messages from a particular user, enter the DIRECTORY/FROM_SUBSTRING command.

For example, DIR/FROM=SYSTEM would list all messages sent from the SYSTEM account.

Displaying a List of All Your Folders

To display a listing of all folders, use the DIRECTORY/FOLDER command.

```
MAIL> DIRECTORY/FOLDER
Listing of folders in DKA100:[USER1]MAIL.MAI
  Press CTRL/C to cancel listing
AGENDAS                MAIL
MEETINGS                REPORTS
TO_DO
```

MAIL>

Specifying a Folder

DIRECTORY/NEW allows you to list new (unread) messages.

DIRECTORY MAIL allows you to list old (previously read) mail messages.

Which folder am I in when I enter the Mail utility?

- NEWMAIL, if there are any unread messages
- MAIL, if there are no unread messages

Reading Messages

Notification About New Messages

The system notifies you when you have new mail messages.

- When you log in to the system, you are notified of any new mail messages.

```
Welcome to OpenVMS Alpha Operating System Version 7.2
```

```
Username: USER1
```

```
Password:
```

```
Welcome to OpenVMS Alpha Operating System Version 7.2 on node TIDY
```

```
  Last interactive login on Tuesday, 12-JAN-1999 11:20:33.04
```

```
  Last non-interactive login on Thursday, 14-JAN-1999 09:22:41.35
```

```
  You have 1 new Mail message.
```

```
$
```

- If you are already logged in when a message arrives, the Mail utility displays an informational message on your terminal screen.

```
$
```

```
New mail on node TIDY from NEAT::USER2
```

```
$
```

Disabling Notification About New Messages

You will not be notified when a new mail message arrives if you enter either of the following DCL commands:

- `$ SET BROADCAST=NOMAIL`
- `$ SET TERMINAL/NOBROADCAST`

`SET BROADCAST/NOMAIL` is preferred because it does not intercept all system messages, just those from Mail. If you use the `SET TERMINAL/NOBROADCAST` command, you will also miss broadcast messages from the system manager.

READ Command

Enter the READ command to read messages in the Mail utility.

MAIL> READ

To read the first new message:

- Invoke the Mail utility.
- Press **[Return]** in response to the MAIL> prompt. (READ is the default command.)
(or)
Issue the READ command

To read a message you receive while using the Mail utility, enter the READ/NEW command.

Sample Read Operations

You can type a variety of commands at the MAIL> prompt to read a mail message.

To read...	Enter...
The message associated with message number <i>n</i>	READ <i>n</i>
The next message	READ
New messages that arrived while you are in MAIL	READ/NEW
The message having the next-highest message number	NEXT
The message having the lowest message number	FIRST
The message having the highest message number	LAST
The current (last-read) message	CURRENT
The message preceding the current message	BACK

Controlling Display of Screen Information

The /PAGE[=keyword] qualifier controls the display of information on the screen. It is valid with all the read operations and the DIRECTORY commands.

You can use the following keywords with the /PAGE qualifier:

- CLEAR_SCREEN - clears the screen before each screen is displayed.
- SCROLL - displays information one line at a time.
- SAVE[=*n*] - enables screen navigation of information, *n* being the number of pages to store.

This qualifier allows you to navigate through screens of information. It stores up to 5 screens and up to 255 columns of information.

Example: Reading a Mail Message

```
❶ $ MAIL
❷ You have 1 new message
MAIL> ❸
❹ #1                               ❺ 14-JAN-1999 09:19:25.33           ❻ NEWMAIL
From:      TIDY::USER3
To: USER 1
Subj:      Status Meeting

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

Thanks!

MAIL>
```

Example Notes: Reading a Mail Message

- ❶ Invoke the Mail utility.
- ❷ Message from the Mail utility to tell you that you have unread messages.
- ❸ At the Mail utility prompt, press **[Return]** to read the first message. (Same as entering READ.)
- ❹ This is the number of the message that you are reading.
- ❺ The time and date that you received this message.
- ❻ The current folder.
- ❼ The message.

Sending Messages

Ways to Send a Message

There are three ways to send a message:

- Create the message and send it.
- Read a message and then forward it to another user.
- Reply to a message you have received.

SEND Command

Use the SEND command to send a message to another user.

MAIL> SEND

You must supply the following information:

- The node (if different from your node) and user name of the recipient(s) in the form **NODE::USER**.
- The subject of the message.
- The message itself. Press **[Return]** at the end of each line.

There are two ways to terminate the message:

- Press **[Ctrl/Z]** after the last line.
- Press **[Ctrl/C]** to cancel the message.

Editing Your Message

SEND/EDIT allows you to edit your message. The message will be sent when you exit from the editor.



Note

Why should you include /EDIT when you are sending an existing file? Perhaps you want to send someone a piece of a command procedure to show how to accomplish a particular task. You don't need to send the whole command procedure, so you edit out the parts that are not needed. The file on disk is not modified.

EVE is the default editor. To change your default editor, enter the SET EDITOR command. This setting remains in effect even after you exit the Mail utility and end this terminal session.

```
MAIL> SET EDITOR EDT
```

Example: Sending a Mail Message

In the following example, the user sends a short mail message to another user.

```
❶ $ MAIL
❷ MAIL> SEND
❸ To: TIDY::USER7
❹ Subj: Project meeting
Enter your message below. Press [Ctrl/Z] when complete, or [Ctrl/C] to quit:
❺ Today's meeting is at 1:00 in the Zuse conference room.
    See you there!
❻ [Exit]

MAIL> EXIT
$
```

Example Notes: Sending a Mail Message

- ❶ Invoke the Mail utility.
- ❷ Issue the SEND command. The default is SEND/NOEDIT. If you enter SEND/EDIT, you can edit the message.
- ❸ Specify the node and user to receive the message.
- ❹ Specify a subject for the message. (FYI is not a helpful subject for a mail message.)
- ❺ This is the text of the message.
- ❻ When you press [Ctrl/Z] to send your message, it echoes on your screen as [Exit]. [Ctrl/C] echoes as [Cancel].

Sending a File as a Message

The Mail SEND file-spec command allows you to send the contents of a file.

MAIL> SEND file-spec

The default file type is TXT, as shown in the error message below. When the user tried to send a file named ABC, with no file type, it was not found because MAIL searched for a file named ABC.TXT.

```
%MAIL-E-OPENIN, error opening DKA100:[USER]ABC.TXT; as input
-RMS-E-FNF, file not found
```

Copying People on Your Messages

SEND/CC_PROMPT allows you to send copies of a message to other users.

MAIL> SEND/CC_PROMPT

You will be prompted for their names.

Using a Distribution List to Send a Message

A distribution list is a file that contains a list of users and their node names. You must use a text editor to create distribution lists. Distribution lists are not created within the Mail utility.

To create a distribution list:

- Use a text editor to create a distribution list file with the file type .DIS.
- Type one user name per line in the file.
- To include the names of other distribution lists in the file, specify an at sign (@) followed by the name of the distribution list.

To send mail to a distribution list, specify an at sign (@) followed by the name of the distribution list at the To: prompt. For example:

```
MAIL> SEND
To:    @FRIENDS
```

FORWARD Command

Use the FORWARD command to send the current (last read) message to another user.

MAIL> FORWARD

You will be prompted for recipient and subject.

Example: Forwarding a Mail Message

In the following example, the user forwards a mail message to two other users.

```
$ MAIL
```

```
You have 1 new message.
```

```
❶ MAIL>
    #1      14-JAN-1999 10:15:56.59      NEWMAIL
    From:    NEAT::USER2
    To:      TIDY::USER1
    CC:
    Subj:    Templates

           I put the templates for the course files in WORK7:[USER.BUILD]

❷ MAIL> FORWARD
❸ To:  USER3,SLOPPY::USER17
❹ Subj: Course file templates

MAIL>
```

Example Notes: Forwarding a Mail Message

- ❶ Press **[Return]** to read your new message.
- ❷ Enter the FORWARD command to route the message to another user. The default is FORWARD/NOEDIT. If you enter FORWARD/EDIT, you can edit the message.
- ❸ Specify the recipient(s) of the forwarded message.
- ❹ Specify the subject of the forwarded message.

Using a Signature File

A signature file (.SIG file) can be used with the mail utility. This appends text at the end of a mail message, often formatted as a user's business card.

You can include a signature file automatically (by default) with every mail message or you can override the default by including different signature files.

To include a signature file by default, use the SET SIGNATURE_FILE command:

```
MAIL> SET SIGNATURE_FILE MY_ADDRESS.SIG
```

To disable the default signature file, type:

```
MAIL> SET NOSIGNATURE_FILE
```

To temporarily override an existing signature file, specify the /SIGNATURE_FILE=file-name qualifier with the DCL command MAIL or with any of the following Mail commands:

- SEND
- MAIL
- REPLY
- FORWARD

To temporarily exclude an existing signature file from a mail message, specify the /NOSIGNATURE_FILE qualifier.

REPLY Command

Use the REPLY command to respond to the sender of the current message.

MAIL> REPLY

Use REPLY the same way you use the SEND command, except:

- You will not be prompted for the recipient.
- The subject will be the same as that of the original message, preceded by RE:

The ANSWER command is the same as REPLY.

Example: Reading and Replying to a Mail Message

In the following example, the user reads a new message and replies to it.

```
$ MAIL
You have 1 new message.
❶ MAIL>
   #1      15-JAN-1999 10:15:04.20      NEWMAIL
From:  TIDY::USER2
To:    NEAT::USER1
CC:
Subj:  Project plan
Do you have a current copy of the project plan?
❷ MAIL> REPLY
❸ To:  TIDY::USER2
❹ Subj: RE: Project plan
Enter your message below. Press [Ctrl/Z]when complete, or [Ctrl/C]to quit:
❺ Yes, I'll make a copy for you. [Exit]
MAIL>
```

Example Notes: Reading and Replying to a Mail Message

- ❶ Press **[Return]** to read your new message.
- ❷ Enter the REPLY command to respond to the message. The default is REPLY/NOEDIT. Enter REPLY/EDIT if you wish to edit your response. Enter REPLY/EDIT/EXTRACT to use the text of this message in your response.
- ❸ The Mail utility supplies the name of the recipient of the reply. You can ask to be prompted for additional recipients by entering REPLY/CC_PROMPT.
- ❹ The Mail utility supplies the subject of the reply. Specify a different subject by entering REPLY/SUBJECT="New subject".
- ❺ When you press **[Ctrl/Z]** to send your reply, it echoes on your screen as **[Exit]** . If you press **[Ctrl/C]** to cancel your reply, it echoes on your screen as **[Cancel]** .

Modifying the Action of the REPLY Command

You can use qualifiers to modify the action of the REPLY command. Some commonly used qualifiers are listed below.

If you want to...	Use this command:
Route your reply to other users	REPLY/CC_PROMPT
Send a copy of your reply to yourself	REPLY/SELF
Specify a different subject for your reply	REPLY/SUBJECT="new subject"
Use the editor to create your reply	REPLY/EDIT

Deleting Messages

DELETE Command

Use the DELETE command to remove a mail message from your current mail folder.

MAIL> DELETE

Either a single message or a range of messages can be deleted.

If a message number is omitted, this command marks the message you are currently reading for deletion.

How a Deleted Message Is Handled

The Mail utility automatically moves a message to the WASTEBASKET folder when you issue the DELETE command. Deleted messages can be read, printed, forwarded, and replied to while they are in the WASTEBASKET folder.

When you exit from the utility or change mail files, the WASTEBASKET folder empties automatically if AUTO_PURGE is set.

When you quit from the utility, the WASTEBASKET folder is not emptied. The deleted messages will still be in the WASTEBASKET folder when you return to Mail.

Enter the PURGE command to empty the WASTEBASKET folder without exiting from the Mail utility.

Sample DELETE Commands

If you want to delete...	Enter:
The message you have just read	MAIL> DELETE
The specified message in the current folder (in this case message number 3)	MAIL> DELETE 3
A range of messages (in this case, message numbers 1, 3, 5, 6, 7, 9, and 11)	MAIL> DELETE 1,3,5-7,9:11

Retrieving a Deleted Message

You can retrieve a message from the WASTEBASKET folder by reading it and moving it to another folder. This is only possible if the WASTEBASKET folder has not been emptied.

Example: Deleting and Retrieving a Mail Message

```

❶ MAIL> DIRECTORY
                                     MAIL
# From          Date      Subject

1 TIDY::USER1    11-JAN-1999 Completed chapters
2 NEAT::USER2    14-JAN-1999 Project Status
❷ MAIL> DELETE 1
❸ MAIL> DIRECTORY
                                     MAIL
# From          Date      Subject
1 (Deleted)
2 NEAT::USER2    14-JAN-1999 Project Status
❹ MAIL> DIRECTORY WASTEBASKET
                                     WASTEBASKET
# From          Date      Subject
1 TIDY::USER1    11-JAN-1999 Completed chapters
❺ MAIL> 1
#1      11-JAN-1999 10:50:53.34      WASTEBASKET
      From:  TIDY::USER1
      .
      .
      .
❻ MAIL> MOVE PROJECT

MAIL>

```

Example Notes: Deleting and Retrieving a Mail Message

- ❶ List the mail messages in the MAIL folder.
- ❷ Delete message number 1.
- ❸ Note that message number 1 is marked as deleted.
- ❹ List the mail messages in the WASTEBASKET folder.
- ❺ Read the deleted message.
- ❻ Move the deleted message to another folder called PROJECT. This message is no longer marked for deletion.

PURGE Command

Use the PURGE command to delete all messages from the WASTEBASKET folder.

MAIL> PURGE

When you exit from Mail, an implicit purge is done to empty the WASTEBASKET folder.



Note

The wastebasket will be automatically purged only if the SET AUTO_PURGE command has been entered. Note also that if AUTO_PURGE is set, the WASTEBASKET folder is also purged when a user switches files using the SET FILE command.

Additional Mail Features

Other Features

The Mail utility has many features you can use to customize its use. The following is a brief list of topics you might wish to investigate in either the online Help facility or the OpenVMS documentation.

The Mail utility allows you to:

- Use personal folders to organize messages
- Print mail messages
- Extract mail messages into files
- Establish default values in your Mail profile to customize your use of Mail

Multipurpose Internet Mail Extension (MIME) Utility

Purpose

The Multipurpose Internet Mail Extension (MIME) utility is used to compose and decode MIME-encoded mail messages and attachments. MIME messages are non-text messages such as graphics or sound files that may be sent or received using the OpenVMS Mail utility.

MIME will only open text files. Extract the MIME-encoded message into a text file using Mail first.

```
MAIL> EXTRACT/NOHEADER HORSE.TXT
```

If it is not created, you may create a command which invokes the MIME executable using the following command:

```
MIME ::= $SYS$SYSTEM:MIME.EXE
```

To create a MIME file:

```
❶ $ RUN SYS$SYSTEM:MIME.EXE
❷ MIME> ADD HORSE.GRA/TEXT
❸ MIME> SAVE
❹ MIME> LIST/FULL
❺ MIME> EXIT
```

- ❶ Invoke the MIME utility.
- ❷ Add a new body part or attachment to the message being edited. Qualifiers are: /AUDIO, /CONTENT-TYPE= TYPE, ENCODING= , /IMAGE, /MESSAGE, /NAME=file-name, /TEXT, /VIDEO.
- ❸ Writes the message to a file.
- ❹ Displays information about the current message including a list of body parts and attributes.
- ❺ Exit the MIME editor, saving any work in progress.

For more information on MIME, see *OpenVMS User's Manual*

Summary

Concepts

Using the Mail Utility

- Use the MAIL command to invoke the OpenVMS Mail utility.
- All mail messages are organized into folders.
- Exit from the Mail utility by entering the EXIT or QUIT command, or by pressing **[Ctrl/Z]**.
- To display a numbered list of your mail messages, use the Mail DIRECTORY command.
- To read your mail messages, use the READ command.
- To send a message to other users, use the SEND, FORWARD, or REPLY command.
- To remove mail messages, use the DELETE command.
- The Mail utility has many features you can use to customize its use.
- The Multipurpose Internet Mail Extension (MIME) is used to attach nontext files to mail messages.

Commands

Using the Mail Utility

MAIL

DCL command to invoke the Mail utility.

EXIT

Exit from the Mail utility and empty the wastebasket folder.

QUIT

Exit from the Mail utility without emptying the wastebasket folder.

DIRECTORY

List the available messages.

DIRECTORY/FOLDER

List folder names in your mail file.

READ

Display the contents of a message.

FORWARD

Send a copy of an existing message to other users.

REPLY or ANSWER

Send an answer to a message already received.

SEND or MAIL

Send a new message to one or more users.

DELETE

Move a message to the wastebasket folder for deletion.

PURGE

Empty the wastebasket folder.

Using the MIME Utility

ADD

Adds a new body part or attachment to the Mail message being edited.

SAVE

Writes the Mail message with MIME attachments to a file.

LIST/FULL

Displays information about the current message.

EXIT

Exit the MIME utility.

Getting Help

Module 7

About This Module

Introduction

To obtain assistance you may talk to your system manager or an experienced user. However, there are two means of obtaining online assistance:

- A complete documentation set
- An online HELP facility

Objectives

To learn more about OpenVMS features, a user should be able to:

- Use online help.
- Describe the organization of the OpenVMS documentation set.

Resources

- *OpenVMS DCL Dictionary*
- *OpenVMS User's Manual*
- *OpenVMS Command Definition, Librarian, and Message Utilities Manual*
- *OpenVMS System Messages: Companion Guide for Help Message Users*

Using Online Help

Overview

Any user who is logged in to an OpenVMS system can obtain online information about the syntax and usage of any DCL command by invoking the OpenVMS Help facility. Using the Help facility, users can ask very general questions and let the facility prompt them with lists of possible topics; or users can enter very specific requests that detail an exact command, qualifier, and parameter.

This section discusses:

- OpenVMS Help facility
- Help Message utility

OpenVMS Help Facility

Overview

The simplest method of invoking the Help facility is to type the DCL HELP command at the OpenVMS prompt.

```
$ HELP
```

HELP will then display a short description of itself and a list of available topics. Topics that are all in uppercase letters are DCL commands. General topics are in lowercase letters.

Selecting Topics in HELP

To select a topic from the topic list:

- Enter the HELP command.
- Select a topic from the list and type it after the Topic? prompt.
- Select a subtopic and type it after the Subtopic? prompt.

You can also type INSTRUCTIONS for more detailed instructions on how to use HELP or type HINTS if you are not sure of the name of the command or topic for which you need help.

You can abbreviate any topic name, although ambiguous abbreviations result in all matches being displayed.

To return to a previous list of subtopics, or to return to the original listing of topics, use the following rules:

- Press **[Return]** to list the [sub]topics on the previous level.
- Type a question mark (?) to redisplay the most recently requested text.

Ending a Help Session

To end your help session:

- Press **[Ctrl/Z]** to exit from the Help facility. This key combination echoes on your screen as **[Exit]**.

Or

- Press the **[Return]** key one or more times until the DCL \$ prompt is displayed.

Example: Help Session

You can enter the commands in the following example at your terminal as you read through it. Some of the output in the text has been truncated for space reasons. You will see the complete display on your terminal screen.

```
❶ $ HELP
      HELP
      The HELP command invokes the HELP Facility to
      Display information about a command or topic. In
      response to the "Topic?" prompt prompt, you can:
      .
      .
      .
      Additional information available:
      := = @ ACCOUNTING ALLOCATE ANALYZE APPEND
      ASSIGN ATTACH AUTHORIZE AUTOGEN BACKUP CALL CANCEL
      .
      .
      .
❷ Topic? SHOW
      SHOW
      Displays information about the current status of
      a process, the system, or devices in the system.
      Format
      SHOW option
      Additional information available:
❸ Description ACCOUNTING ACL AUDIT BROADCAST
      .
      .
      .
❹ SHOW Subtopic? SYSTEM
      SHOW

      SYSTEM
      Displays status information about current
      Processes.
      Format
      SHOW SYSTEM
      Additional information available:
      Qualifiers
      /BATCH /CLUSTER /EXACT /FULL /HEADING /HIGHLIGHT /IDENT
      .
      .
      .
❺ SHOW SYSTEM Subtopic?
❻ SHOW Subtopic? ?
      SHOW
      Displays information about the current status of
      a process, the
      .
      .
      .
❼ SHOW Subtopic? [Exit]
      $
```

Example Notes: Help Session

- ❶ Invoke the Help facility by typing the HELP command.
- ❷ Select a topic from the list.
- ❸ A topic printed in lowercase is a general topic, not an actual command.
- ❹ Select a subtopic.
- ❺ Press **[Return]** to back up a level.
- ❻ Press ? to see the list of [sub]topics again.
- ❼ Press **[Ctrl/Z]** to exit from the Help facility.

Selecting a Topic from the DCL Command Line

In addition to selecting a topic from those available, you can select one from the DCL command line by typing the topic immediately after the HELP command.

```
$ HELP SHOW
```

If you know more specifically your information needs, you can specify further information on the command line.

```
$ HELP SHOW USERS
```

The information you enter following the HELP command can be anything that can follow the command you are looking up. For example, the following is also a correct HELP command:

```
$ HELP SHOW USERS/FULL
```

Faster Use of Help

To suppress the introductory display on how to use the Help facility, use the /NOINSTRUCTION qualifier.

```
$ HELP/NOINSTRUCTION
```

Help Message Utility

HELP/MESSAGE Command

Use the Help Message utility to get online help for system messages. To display information on how the last command completed, enter:

```
$ HELP/MESSAGE
```

You can also display information about a specific message by including the message identifier or words from the message text. For example:

```
$ HELP/MESSAGE BADACP
```

Using Status Codes

Entering the message status code can also access a message and its description. For example:

```
$ HELP/MESSAGE/STATUS=%X00038090
```

If you do not know the message status code, you can view it by entering the command `SHOW SYMBOL $STATUS` followed by the `$STATUS` global symbol. For example:

```
$ SHOW SYMBOL $STATUS
$STATUS == "%X00038090"
```

Resources

For information on how to use the Help Message utility, see *OpenVMS System Messages: Companion Guide for Help Message Users*.

Using the OpenVMS Documentation Set

Overview

As you work with the OpenVMS operating system, you may have questions that are not answered in sufficient detail by online help. Or, you may not be able to formulate your question in such a way as to use online help to get the answer. In either of these situations, you will want to consult the OpenVMS documentation.

This section discusses:

- Components of OpenVMS documentation

Components of OpenVMS Documentation

Three Sets of Documentation

The OpenVMS documentation set is divided into three sets of documentation:

This set...	Contains:
Base Documentation Kit	Information on the activities you are most likely to perform as a user or system manager of a small standalone system, system security information, no programming information
Standard Documentation Set	Base Documentation Kit plus: ■ Master Reference Kit ■ Advanced System Management Kit ■ Networking Kit ■ Programming Kit
Standard + Optional Documentation Set	Standard Documentation Set and all optional documentation (except for System Integrated Products documentation)

See the *Overview of OpenVMS Documentation* for complete titles of OpenVMS documents.

Who Should Use Each Set of Documentation

If you are a...	You should use:
User of a small standalone system or a large OpenVMS system	Base Documentation Kit
System manager of a small standalone system	Base Documentation Kit
System manager of a large OpenVMS system or an OpenVMS cluster system	Standard Documentation Set
System or application programmer	Standard Documentation Set

Documentation for OpenVMS Alpha and OpenVMS VAX

Most manuals contain information common to both OpenVMS Alpha and OpenVMS VAX systems. Any functional differences between the two are clearly described in the text.

A small number of books has been produced specifically for OpenVMS Alpha or OpenVMS VAX systems. For example, because the command displays and crash analysis sessions for the system dump analyzer are different on Alpha and VAX systems, two separate manuals have been produced.

OpenVMS Master Index

The *OpenVMS Command Definition, Librarian, and Message Utilities Manual* is a compilation of the individual indexes for the books in the Standard Documentation Set.

OpenVMS Glossary

The *OpenVMS DCL Dictionary* defines terms specific to the OpenVMS operating system that are used in the OpenVMS documentation.

The terms are presented in alphabetical order. They have not been grouped into categories.

Summary

Concepts

Using Online Help

- You can obtain online information about the syntax and usage of any DCL command by invoking the OpenVMS Help facility.
- You can obtain online information about system messages by using the Help Message utility.

Using the OpenVMS Documentation Set

- The OpenVMS documentation set is composed of three sets of documentation: the Base Documentation Kit, the Standard Documentation Set, and a combination of the Standard Documentation Set and optional documentation.

Commands

Using Online Help

HELP

Invokes the OpenVMS Help facility to obtain online help.

HELP/MESSAGE

Invokes the Help Message utility to obtain online help about system messages.

Protecting Your Data

Module 8

About This Module

Introduction

One of the most important things a user needs to learn is how to protect data from unauthorized access or accidental deletion.

This module describes how to display and control file protection, how to erase files that are no longer needed, and gives reasons why files should be archived or backed up regularly.

Objectives

To perform basic data protection tasks, a user should be able to:

- Use UIC-based protection to protect files and directories.
- Save copies of files.

Resources

- *OpenVMS DCL Dictionary*
- *OpenVMS System Management Utilities Reference Manual*
- *OpenVMS User's Manual*
- *Security Guide*

Using File Protection

Overview

To prevent other users from accessing your files, you can protect them using UIC protection codes. A further level of protection can be added through the use of access control lists (ACLs). Typically, users only make use of UIC-based file protection.

This section discusses:

- Security profiles
- User identification codes, UICs
- Protection codes
- Manipulating protection codes

Security Profiles

Protected Objects

The OpenVMS operating system controls access to any object that contains shareable information. These objects are called protected objects. A file is an example of a protected object.

Every protected object has a set of access requirements specifying who has a right to access the object in a given manner.

Contents of a User Security Profile

User processes and applications have security profiles, which contain the following information:

- User identification code (UIC) identifying the user
- Rights identifiers held by the process
- Privileges, if any

Contents of an Object's Security Profile

The security elements of any object comprise its security profile.

An object's security profile contains the following information:

- UIC of the owner of the object
- Protection code
- Access control list (ACL), if any

This information determines what users can access the object, and what types of access are permitted.

User Identification Codes, UICs

Purpose of UICs

The OpenVMS operating system uses a file's UIC and protection code to determine who can:

- Read the file
- Write the file
- Execute the file
- Delete the file

UIC Components

Every user on an OpenVMS system has a user identification code (UIC) that identifies the user to the system. A UIC has two parts.

[group,member]

Several users may belong to the same group, but they should have different member numbers.

UIC Formats

A UIC has either a **numeric** or **alphanumeric** format. The numeric format of a UIC is a pair of numbers in brackets, as follows:

- The group number can range from 1 to 37776 (octal).
- There can be from 0 to 177776 (octal) members in each group.
- Example of a numeric UIC:

[100,30]



Note

Compaq reserves group 1 and groups 300-377 for its own use.

The alphanumeric format of the UIC is easier for users to remember.

[HPMR_GROUP,USER1] or [USER1]

Example: Displaying Your UIC

Use the SHOW PROCESS command to display the UIC assigned to your account.

```
$ show process
```

```
16-MAR-1999 14:47:03.59  User: PANGAKIS      Process ID: 4EE00898
                          Node: DELPHI       Process name: "PANGAKIS"
```

```
Terminal:      RTA6: (SPARTA::PANGAKIS)
User Identifier: [VMS,PANGAKIS]
Base priority: 4
Default file spec: WORK45:[PANGAKIS]
```

```
Devices allocated: DELPHI$RTA6:
```

Example: Displaying the UIC of a File

When a file is created, the OpenVMS operating system assigns a UIC to the file. The file's owner is the creator of the file, unless otherwise specified. Use the DCL DIRECTORY command with the /OWNER qualifier to see the UIC attached to a file.

```
$ DIRECTORY/OWNER MY_FILE.DAT
```

```
Directory WORK3:[USER1]
```

```
MY_FILE.DAT;1      [12,200]
```

```
Total of 1 file.
```

```
$
```

Protection Codes

Types of Access

Protection codes describe the kind of access permitted to a file, directory, or other object.

Types of File Access

UIC-based protection allows four types of file access.

This protection code...	Allows this kind of access...
Read (R)	COPY, TYPE, PRINT, DIRECTORY
Write (W)	EDIT (modify)
Execute (E)	RUN
Delete (D)	DELETE, PURGE, RENAME

All protected objects also support control access, which allows a user to examine and modify the security elements (ACL, protection code, and UIC). Control access is explicitly stated in an ACL but never appears in the UIC-based protection code.

User Categories

An OpenVMS system recognizes four categories of users based on the UIC of the file or object to be accessed.

- **SYSTEM (S)** is a special category given to users who need special privileges to perform system management tasks.
- **OWNER (O)** is the user with the UIC that matches the UIC on the file. Usually the owner is the user who created the file.
- **GROUP (G)** includes all users who belong to the same group as the owner.
- **WORLD (W)** includes all users with access to the system. The other categories are subsets of WORLD.

If the access requested is granted to any category to which the requesting process belongs, the access will be allowed. For example, if a group member requests to read the file, the request will be granted because all group members belong to the category WORLD and that category has read access to the file.

Assigning UICs to Categories

When a user requests access to a file or a directory, the system compares the UIC of the file or directory with the UIC of the user requesting access.

Category	Group Number	Member Number
SYSTEM	0 through 10 (octal)	Does not matter
OWNER	Matches group number of file UIC	Matches member number of file UIC
GROUP	Matches group number of file UIC	Does not matter
WORLD	Does not matter	Does not matter

Figure: Elements of a Protection Code

The following figure shows the elements of a protection code, as it would be specified by a SET SECURITY/PROTECTION command.

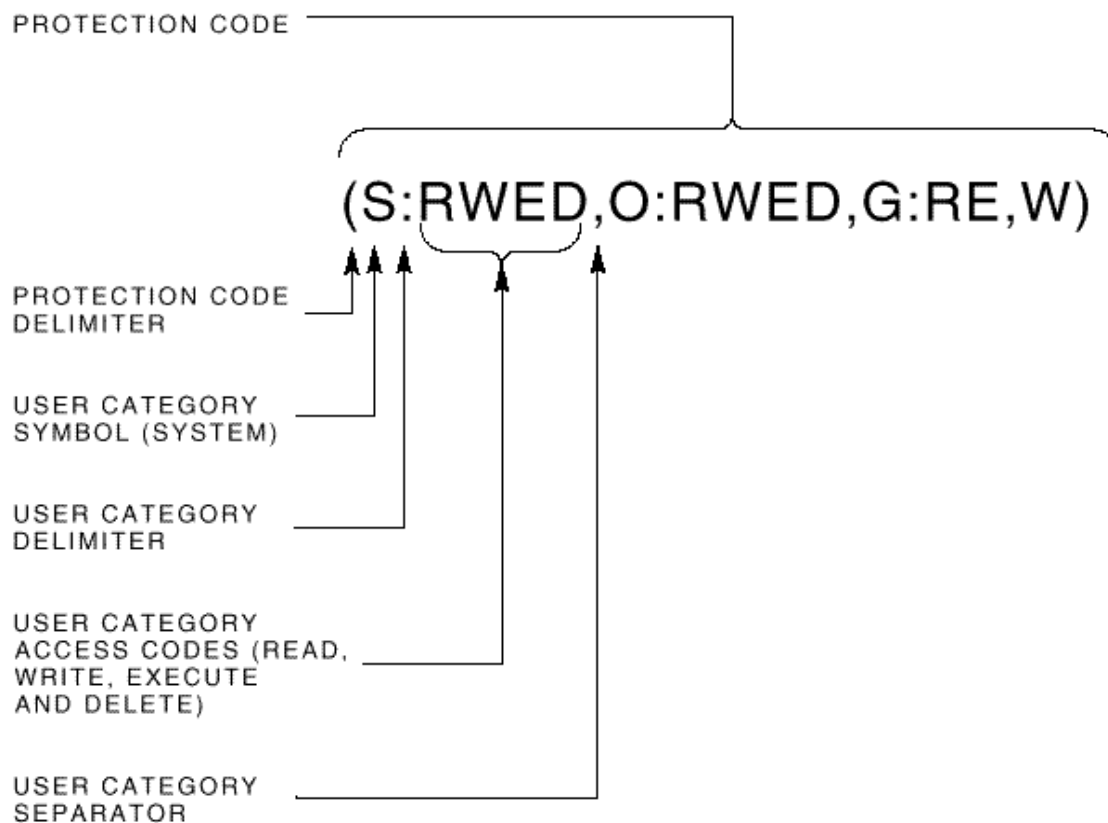


Figure Notes: Elements of a Protection Code

- SYSTEM and OWNER are allowed to read, write, execute, and delete the file.
- GROUP members are allowed to read and execute the file.
- WORLD members have no access.

Displaying File Security Information

Use the DIRECTORY command with various qualifiers to display security information for a file.

To display this information...	Use this command:
UIC protection code	DIRECTORY/PROTECTION
Owner	DIRECTORY/OWNER
Access control list (if any)	DIRECTORY/ACL
All of the above	DIRECTORY/SECURITY

The SHOW SECURITY command displays the same information as DIRECTORY/SECURITY:

SHOW SECURITY filespec

Example: Displaying File Protection

The following example uses various DIRECTORY qualifiers to display a file's protection.

```
$ DIRECTORY/PROTECTION SAMPLE.FILE

Directory WORK3:[USER1]

SAMPLE.FILE;2          (RE,RWED,RE, )

Total of 1 file.
$ DIRECTORY/OWNER SAMPLE.FILE

Directory WORK3:[USER1]

SAMPLE.FILE;2          [12,200]

Total of 1 file.
$ DIRECTORY/ACL SAMPLE.FILE

Directory WORK3:[USER1]

SAMPLE.FILE;2

Total of 1 file.
$ DIRECTORY/SECURITY SAMPLE.FILE

Directory WORK3:[12,200]

SAMPLE.FILE;2          [USER1]          (RE,RWED,RE, )

Total of 1 file.
$
```

Note that SYSTEM and GROUP have read and execute access to the file, OWNER has read, write, execute, and delete access, and WORLD has no access. There is no access control list for this file.

The following example shows the output from the SHOW SECURITY command:

```
$ SHOW SECURITY SAMPLE.FILE

WORK3:[USER1]SAMPLE.FILE;2 object of class FILE
  Owner: [12,200]
  Protection: (System: RE, Owner: RWED, Group: RE, World)
  Access Control List: <empty>
$
```

Default Protection

Your top level directory has a default file protection code associated with it. When you create a file in your top level directory or any subdirectory, the OpenVMS operating system applies the default protection to the new file unless you specify an alternate protection code. You can specify a different protection code for a file when you create the file or you can change the protection on a file after it is created.

The default protection is not the same as the protection code on the directory. Default protection is the protection code that will be applied to new files created in the directory or in any subdirectories unless the creator has specifically designated a protection code for the directory. The protection code on the directory determines what user categories can access the directory.

Example: Displaying Default Protection

Use the SHOW PROTECTION command to see the default protection for your process. This is the protection code that will be assigned to any new files you create, unless you specify a different protection code for them.

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

Example: Changing Default Protection

Use the SET PROTECTION command with the /DEFAULT qualifier to change the default protection for your process.

The following example changes the default protection to allow SYSTEM read, write, and execute access and allow GROUP no access.

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

When changing a protection code, you need only enter the portion of the code that you wish to change. Because the OWNER and WORLD categories were not included in the SET PROTECTION command, their access is unchanged.

Example: Effect of Changing Default Protection

Changing the default protection will only affect the protection code on new files you create. New versions of any file that existed before the directory protection was changed will take the protection code of the previous version of the file.

```

❶ $ SHOW PROTECTION
    SYSTEM=RWED, OWNER=RWED, GROUP=RE, WORLD=NO ACCESS
❷ $ DIRECTORY/OUTPUT=DIRECTORY1.LIS
❸ $ DIRECTORY/PROTECTION DIRECTORY1.LIS
    Directory WORK3:[USER1]
❹ DIRECTORY1.LIS;1      (RWED,RWED,RE, )
    Total of 1 file.
❺ $ SET PROTECTION=(S:R,G:R)/DEFAULT
❻ $ SHOW PROTECTION
    SYSTEM=R, OWNER=RWED, GROUP=R, WORLD=NO ACCESS
❼ $ DIRECTORY/OUTPUT=DIRECTORY2.LIS
❽ $ DIRECTORY/PROTECTION DIRECTORY *.LIS
    Directory WORK3:[USER1]
    DIRECTORY1.LIS;1      (RWED,RWED,RE, )
❾ DIRECTORY2.LIS;1      (R,RWED,R, )

    Total of 2 files.
$
```

Example Notes: Effect of Changing Default Protection

- ❶ Display the current default file protection.
- ❷ Create a new file.
- ❸ Display the protection on the new file.
- ❹ The new file has the default protection.
- ❺ Change the default file protection so SYSTEM and GROUP have only read access.
- ❻ Display the new default file protection.
- ❼ Create another file.
- ❽ Display the protection on both files.
- ❾ The second file was created with the new protection.

Manipulating Protection Codes

Setting File Protection

Use the SET SECURITY/PROTECTION command to control a file's UIC protection code.

Format:

SET SECURITY/PROTECTION=(code)

Example: Setting File Protection

The following command sets a file's protection to allow:

- SYSTEM and OWNER read, write, and execute access
- GROUP and WORLD no access

```
$ SET SECURITY/PROTECTION=(S:RWE,O:RWE,G,W) SAMPLE.FILE
```

Protection on Directory Files

To guard against accidental deletion, directory files are created with a different protection than that given to other files. By default, delete access is denied to all user categories.

```
$ DIRECTORY/PROTECTION COMMANDS.DIR
Directory WORK3:[USER1]
COMMANDS.DIR;1      (RWE,RWE,RE,)
Total of 1 file.
$
```

You can change the protection on a directory file the same way you change the protection on any other file.

Steps in Deleting a Directory File

Because directories are protected against deletion, special steps must be taken when you wish to delete a directory file.

Follow these steps to delete a subdirectory:

- Remove all files from the directory, either by deleting them or renaming them to another directory.
- Set your default to a higher level directory.
- Change the protection on the directory file so that you (the owner) can delete it.
- Delete the directory file.

Example: Deleting a Subdirectory

The following example shows the steps in deleting a subdirectory.

```
❶ $ DIRECTORY [USER1.DOC]
Directory WORK3:[USER1.DOC]
CLASS.LIST;4          CLOCK.EXE;1      COLOR.COM;4          DEG.EXE;1
JOE_EVE.TPU$SECTION;1 MYFILE.TXT;1          NOTE.COM;4
REMIND.EXE;1          REMLOG.EXE;1      TRNG.PLAN;6          VT100.CLR;1
Total of 11 files.
❷ $ DELETE [USER1.DOC]*.*;*
❸ $ DIRECTORY [USER1.DOC]
%DIRECT-W-NOFILES, no files found
❹ $ SET SECURITY/PROTECTION=0:RWED DOC.DIR
❺ $ DELETE DOC.DIR;1
❻ $ DIRECTORY DOC.DIR
%DIRECT-W-NOFILES, no files found
❼ $ DIRECTORY [.DOC]
%DIRECT-E-OPENIN, error opening WORK3:[USER1.DOC]*.*;* as input
-RMS-E-DNF, directory not found
-SYSTEM-W-NOSUCHFILE, no such file
$
```

Example Notes: Deleting a Subdirectory

- ❶ List the names of any files in the subdirectory you wish to delete.
- ❷ Delete the files that are in the subdirectory. (You might prefer to rename them to another directory, rather than delete them.)
- ❸ Verify that the subdirectory contains no files.
- ❹ Assuming you are the owner of the file, give yourself delete access to the subdirectory file.
- ❺ Delete the directory file.
- ❻ The directory file no longer exists.
- ❼ The subdirectory is no longer part of your directory structure.

Common Errors in Deleting Directory Files

Errors occur when you try to delete a directory file that:

- Still catalogs other files
- Is protected against deletion

Common Error: Deleting a Directory That Is Not Empty

You cannot delete a directory file that still catalogs other files.

```
$ DELETE DATA.DIR;*
%DELETE-W-FILNOTDEL, error deleting WORK3:[USER1]DATA.DIR;1
-RMS-E-MKD, ACP could not mark file for deletion
-SYSTEM-F-DIRNOTEMPTY, directory file is not empty
```

What to do:

- Delete all files cataloged by the directory.

```
$ DELETE [USER1.DATA]*.*;*
```

- Then delete the directory file.

Common Error: Deleting a Protected Directory File

You cannot delete a directory if you do not have delete access to it.

```
$ DELETE DATA.DIR;*
%DELETE-W-FILNOTDEL, error deleting WORK3:[USER1]DATA.DIR;1
-RMS-E-PRV, insufficient privilege or file protection violation
```

What to do:

- Change the protection code on the subdirectory to give yourself (the owner) delete access. Enter one of these commands:

```
$ SET SECURITY/PROTECTION=0:RWED DATA.DIR  
$ SET FILE/PROTECTION=0:RWED DATA.DIR
```

Saving Copies of Files

Overview

On a large system, the system manager will establish a schedule for making backup copies of files. If you work on a small standalone system, you may have to back up your own files. You should perform this operation on a regular basis to make sure that you do not lose important files due to system problems or your own actions.

This section discusses:

- Backup utility

Backup Utility

What the Backup Utility Can Do

Use the Backup utility to copy files from disk to disk or from disk to tape and to restore files to disk. You can make copies of:

- Individual files and their locations in a directory structure
- A directory structure and the files in it
- All directory structures and files on a volume

Magnetic tape is frequently used as a storage medium for file copies because it is inexpensive and easy to store and transport.

BACKUP Command

The BACKUP command has the following format:

BACKUP input-specifier output-specifier

You can use several types of qualifiers to modify the default behavior of the BACKUP command.

- Add a qualifier to the BACKUP command to modify the action of the entire command.
- Qualifiers following the input specifier allow you to select specific files for backup and specify how the input is processed.
- The output specifier may also have qualifiers.

Save Sets

The Backup utility stores files and directories in a specially formatted file called a *save set*. The special format makes it possible to save the relationship of a file to a directory structure as well as the contents of the file. For example, the Backup utility can save the fact that FILE.DAT was located in the directory [SMITH] and save FILE.DAT itself.

A save set is a single file that may contain the contents of many files and their associated directory structures. For example, a save set may contain all the files and the directory structures of an entire disk.

You can create a save set file on a disk or tape volume. A save set is a normal OpenVMS system file, with a standard OpenVMS file name. You can copy it, rename it, or delete it using the same DCL commands you use to manage other files.

Example: Making a Save Set of All Files in a Directory

To make a save set of all the files in a subdirectory named WORK:

```
$ BACKUP  [.COMMANDS]*.*  COMMANDS_DIR.BCK/SAVE_SET
```

Use the DIRECTORY command to see the save set created by the command:

```
$ DIRECTORY  COMMANDS_DIR.BCK
```

```
Directory USER_DISK:[USER1]
```

```
COMMANDS_DIR.BCK;1
```

```
Total of 1 file.
```

```
$
```

Example: Listing the Contents of a Save Set

Use the BACKUP/LIST command utility to list the contents of a save set.

```
$ BACKUP/LIST  COMMANDS_DIR.BCK/SAVE_SET
```

```
Listing of save set(s)
```

```
Save set:          COMMANDS_DIR.BCK
Written by:        USER1
UIC:               [000012,000200]
Date:              14-AUG-1995 14:20:41.72
Command:           BACKUP [.COMMANDS] COMMANDS.BCK/SAVE
Operating system:  OpenVMS AXP version E7.0
BACKUP version:    V6.2
```

```
CPU ID register:   80000000
Node name:         _TIDY::
Written on:        _$1$DUA1:
Block size:        32256
Group size:        10
Buffer count:      28
```

```
[USER1.COMMANDS]DOC.COM;4      20  14-AUG-1995 15:55
[USER1.COMMANDS]GREET.COM;1    2   12-AUG-1995 13:56
[USER1.COMMANDS]LOGICALS.COM;4 15   12-AUG-1995 14:10
[USER1.COMMANDS]LSE_EDIT.COM;1 1   11-AUG-1995 13:56
```

```
.
```

```
.
```

```
.
```

```
Total of 22 files, 114 blocks
```

```
End of save set
```

```
$
```

Restoring a File from a Save Set

To retrieve a file from a save set and restore it to a directory, use the BACKUP command with the save set as the input specifier. If the input specifier is a save set on disk, use the /SAVE_SET qualifier to the input specifier. To restore one file only use the /SELECT qualifier.

```
$ BACKUP/LOG
 _From: COMMANDS.BCK/SAVE/SELECT=TERMINAL.COM
 _To: *.*
%BACKUP-S-CREATED, created USER_DISK:[USER1]TERMINAL.COM;1
$
```

In this example, the /LOG qualifier was used to cause BACKUP to display a message when the operation completed.

Summary

Concepts

Using File Protection

User processes and applications have security profiles, which determine whether they have the authority to access particular objects.

- Objects have security profiles, which specify who can access them and what kind of access is permitted.
- Every file has a user identification code (UIC) which identifies the user who created the file, or owns it.
- Every file has a protection code that establishes the type of access a particular category of user has to the file.
- Use DCL commands to display and alter the protection of your files.

Saving Copies of Files

- Archiving and regular backup of files help to protect data from loss or corruption.

Commands

Using File Protection

SHOW PROCESS

Display process information, including the UIC assigned to your account.

DIRECTORY/PROTECTION**DIRECTORY/OWNER****DIRECTORY/ACL****DIRECTORY/SECURITY****SHOW SECURITY**

Display security information about a file.

SHOW PROTECTION

Display default protection for your process.

SET PROTECTION /DEFAULT

Change the default protection for your process.

SET SECURITY/PROTECTION

Change the protection on a file.

Saving Copies of Files

BACKUP

Runs the Backup utility.

Customizing Your Working Environment

Module 9

About This Module

Introduction

OpenVMS offers a number of features that enable you to customize your working environment for convenience and efficiency.

- The names of commonly used files and directories can be represented by short, meaningful logical names that are easier to type and to remember than full file specifications.
- Symbols may be used as shortcuts for entering frequently used commands.
- Many terminal keys may be redefined to enable you to enter commands and other strings by pressing only one or two keys.
- To have these features available every time you log in to your system, you may create a special command file called LOGIN.COM, which will automatically perform these customization tasks for you every time you login.

This module introduces logical names and symbols, describes how to create keypad definitions, and tells how to create a simple login command procedure.

Objectives

To customize the working environment to match specific working needs, a user should be able to:

- Use logical names.
- Use symbols.
- Use keypad definitions.
- Write a simple command procedure.

Resources

- OpenVMS DCL Dictionary
- OpenVMS User's Manual
- User's Guide for your terminal or workstation

Using Logical Names

Overview

A *logical name* is a character string used to represent other character strings called *equivalence strings*. Once a logical name has been equated to a string, it can be used instead of the equivalence string.

Equating a short logical name to a long file specification will reduce the typing effort and help to eliminate mistakes when the logical name is used with commands.

Using logical names to make programs independent of specific devices saves maintenance costs and makes programs portable to other OpenVMS systems. A program to be run on many different systems may need to access a file located on a different disk from the program. At the time the program is written, the programmer has no idea what disk this will be. If the program is written using a logical name for the disk and directory, the logical name can be defined by the program installation procedure. The installation program will ask the installer to input the disk where the file should be located and then define the logical name to equate to this disk.

This section discusses:

- Creating logical names
- Using logical names
- Determining the equivalence of a logical name
- Removing logical names
- Logical names the system creates for you

**Note**

This section does not discuss making logical names available to multiple users.

Creating Logical Names

Commands Used to Create Logical Names

There are two DCL commands that enable you to create a logical name.

They accomplish the same task, but they use different syntax.

ASSIGN **equivalence-string[,...]** **logical-name[:]**

```
$ ASSIGN DKA100:[CHARLES] CHARLIE
```

(or)

DEFINE **logical-name** **equivalence-string[,...]**

```
$ DEFINE CHARLIE DKA100:[CHARLES]
```

Both commands create logical names that are available only to you. Other users on the system will not be able to use your logical names.

Rules for Creating Logical Names

Logical names may have a maximum of 255 characters. A logical name may contain alphanumeric characters, as well as the underscore (_), dollar sign (\$), and hyphen (-).

The equivalence string may have 255 characters as well. It may contain punctuation marks (colons, brackets, periods) required as part of a complete file specification. It may also contain alphanumeric characters, underscore (_), dollar sign (\$), and hyphen (-).

Do not specify a colon at the end of a logical name.

Logical Names in Equivalence Strings

The equivalence string for a logical name may be another logical name or may contain a logical name.

In the following example the logical name DISK is defined and then used in the equivalence string of the logical name REPORT.

```
$ DEFINE DISK DKA100
$ DEFINE REPORT DISK:[KEES.DATA]FEBRUARY.TXT
```

Duration of a Logical Name

Logical names are not stored permanently on the system. When you create a logical name it is defined for the life of your session. When you log out of the system the logical name is deassigned. If you want a logical name to be available every time you log in, you must create the logical name in a file to be executed each time you log in, possibly LOGIN.COM.

Using Logical Names

Common Uses for Logical Names

Logical names are most commonly used in place of all or part of a file specification. They may also be used to simplify the process of sending mail to other users.

Example: Using a Logical Name for a Directory Specification

One of the most common uses for a logical name is to specify the directory portion of a file specification. In the following example, the logical name WS is equated to a directory specification and then used in a command.

```
$ ASSIGN [SMITH.WORKSHEET] WS
$ DIRECTORY WS

Directory DKA800:[SMITH.WORKSHEET]
.
.
.
```

Example: Using a Logical Name for a Complete File Specification

When using a logical name to refer to a file specification, the logical name may represent the entire file specification or just the leftmost portion of the file specification. In the following example, the logical name FEBRUARY is equated to the complete file specification.

```
$ ASSIGN USER8:[SMITH.WORKSHEET]FEBRUARY_RESULTS.DAT FEBRUARY
$ DIRECTORY FEBRUARY

Directory DKA800:[SMITH.WORKSHEET]

FEBRUARY_RESULTS.DAT;1

Total of 1 file.
$
```

Example: Using a Logical Name in the Left Part of File Specification

A logical name may be the left part of a file specification. The logical name WS is equated to the disk and directory in which a particular data file is located. The logical name is used as the left part of the file specification and separated by a (:) from the file name.

```
$ ASSIGN DKA800:[SMITH.WORKSHEET] WS
$ DIRECTORY WS:FEBRUARY_RESULTS.DAT
```

```
Directory DKA800:[SMITH.WORKSHEET]
```

```
FEBRUARY_RESULTS.DAT;1
```

```
Total of 1 file.
```

```
$
```

Common Error in Using Logical Names

You may not use a logical name as the right-hand portion of a file specification.

```
$ ASSIGN JANUARY_RESULTS.DAT JANUARY
$ DIRECTORY [SMITH.WORKSHEET]JANUARY
%DIRECT-W-NOFILES, no files found
$ DIRECTORY JANUARY
```

```
Directory DKA800:[SMITH.WORKSHEET]
```

```
JANUARY_RESULTS.DAT;1
```

```
Total of 1 file.
```

```
$
```

Using Logical Names with the Mail Utility

- You may use logical names to simplify the process of sending mail to one or more users.
- Assign a short logical name to equate to the full specification of a user on another node to whom you send mail frequently.

```
$ ASSIGN    NEAT::SMITH    DANA
```

Use the logical name as the recipient of a mail message.

```
$ MAIL
```

```
MAIL> SEND
```

```
To:      DANA
```

```
.
```

```
.
```

```
.
```

- There are two ways you may use logical names to send mail to a group of users.
 - You may equate a logical name to a short string that represents several user specifications. When a comma (,) is part of the equivalence, enclose the equivalence string in quotation marks.

```
$ ASSIGN  "NEAT::SMITH,TIDY::STEPHENS,READ"  PROJECT_TEAM
```

```
$ MAIL
```

```
MAIL> SEND
```

```
To:      PROJECT_TEAM
```

```
.
```

```
.
```

```
.
```

- You may also assign a logical name to the file specification for a distribution list.

```
$ ASSIGN  [SMITH.LISTS]SITE_USERS.DIS  SITE
```

```
$ MAIL
```

```
MAIL> SEND
```

```
To:      @SITE
```

```
.
```

```
.
```

```
.
```

Determining the Equivalence of a Logical Name

Translation of Logical Names

When the equivalence string for a logical name is itself a logical name, the translation process may be done in two ways.

- Translate only the first logical name.
- Translate the logical name and a logical name found in the equivalence string, repeating the process for any further logical names found. This is called *iterative translation*.

SHOW TRANSLATION Command

Use the SHOW TRANSLATION command to display the equivalence string of a logical name. If the equivalence string contains a logical name it will be displayed, but not translated further.

SHOW TRANSLATION *logical-name*

Example: SHOW TRANSLATION Command

In the following example, the logical name PAYROLL is assigned another logical name DISK1 in the equivalence string. The SHOW TRANSLATION command only displays the first translation of the logical name.

```
$ ASSIGN DKA100: DISK1
$ ASSIGN DISK1: PAYROLL
$ SHOW TRANSLATION PAYROLL
  PAYROLL = "DISK1:" (LNM$PROCESS_TABLE)
$
```

SHOW LOGICAL Command

The SHOW LOGICAL command iteratively translates the logical name up to 10 levels until all logical names are resolved.

SHOW LOGICAL *logical_name*

Example: SHOW LOGICAL Command

In the following example, the logical name DISK1 is used in the equivalence string for the logical name PAYROLL. Use the SHOW LOGICAL command to display the iterative translation.

```
$ ASSIGN DKA100: DISK1
$ ASSIGN DISK1: PAYROLL
$ SHOW LOGICAL PAYROLL
    "PAYROLL" = "DISK1:" (LNM$PROCESS_TABLE)
1  "DISK1" = "DKA100:" (LNM$PROCESS_TABLE)
$
```

Removing Logical Names

DEASSIGN Command

Use the DEASSIGN command to cancel logical name assignments made with either the ASSIGN or DEFINE commands.

DEASSIGN logical-name

For example, the following command removes the logical name PROJECT_TEAM:

```
$ DEASSIGN PROJECT_TEAM
```

Delete All Process Logical Names

Use the /ALL qualifier to the DEASSIGN command to cancel all of your process logical names.

DEASSIGN/ALL

Logical Names the System Creates for You

Logical Names Created by the System

OpenVMS creates several logical names for your process. Two of these logical names are SYS\$LOGIN and SYS\$LOGIN_DEVICE.

SYS\$LOGIN

When you log in, OpenVMS equates the logical name SYS\$LOGIN to your initial default device and directory. You may think of this directory as your home directory.

You may use the logical name SYS\$LOGIN to return to your home device and directory after you have changed your directory to another device and/or directory.

```
$ SET DEFAULT SYS$LOGIN
```

SYS\$LOGIN_DEVICE

The OpenVMS operating system equates the logical name SYS\$LOGIN_DEVICE to the disk on which your login directory is located.

Use the SHOW DEVICE command with the logical name SYS\$LOGIN_DEVICE to obtain information about your disk.

```
$ SHOW DEVICE SYS$LOGIN_DEVICE
```

Device Name	Device Status	Error Count	Volume Label	Free Blocks	Trans Count	Mnt Cnt
DKA100:(TIDY)	Mounted	0	USER1	167538	2	12

```
$
```



Note

In addition to the above display, a line similar to the following may be displayed if the device is mounted as part of a shadow set:

```
$11$DUA1024: (HSC031) ShadowSetMember 0  
(member of DSA2034:)
```

Using Symbols

Overview

A symbol is a name that represents a numeric, character, or logical value. When you use a symbol in a DCL command line, DCL uses the value you assign to that symbol. By defining a symbol as a command line, you can execute the command by typing only the symbol name.

This section discusses:

- Introduction to symbols
- Creating symbols
- Displaying symbols
- Deleting symbols

Introduction to Symbols

Symbols

Symbols are useful shortcuts for entering frequently used DCL commands, and they may represent data in command procedures.

A symbol may be used:

As a synonym for an entire DCL command

- As a variable within a DCL command procedure

As a parameter passed to a command procedure

The value you assign to a symbol may be made available either locally or globally to DCL.

Local Symbols

Local symbols are usually used as a variable within a command procedure. A local symbol exists:

- During the execution of the command procedure that created it
- In command procedures invoked by the procedure that created it

A local symbol ceases to exist when the procedure that created it exits.

Global Symbols

A *global symbol* applies to all command levels, even after the command procedure that created it has exited. Use global symbols in command procedures when you want the symbol to continue to apply after the command procedure exits.

Duration of Symbols

Symbols are not permanently assigned. A symbol you define during a session goes away when you log out.

You may make a global symbol available to your process each time you log in by placing it in a command procedure to be executed each time you log in, possibly LOGIN.COM.

Data Types for Symbols

Character String

A symbol may represent a character string containing any characters that can be printed. You may include spaces in the string and preserve the use of uppercase or lowercase letters.

Integer

A symbol may represent an integer number. Numbers may be positive or negative. To specify a number in a radix other than decimal (base 10), precede the number (but not the sign) with:

%X for hexadecimal %X27

%O for octal %045

Creating Symbols

Equating Symbols to Character Strings

Use a colon (:) and an equal sign (=) to assign character string values to a symbol.

- Use one colon and one equal sign to create a local symbol:

SYMBOL := STRING

- Use one colon and two equal signs to create a global symbol

SYMBOL ::= STRING

Sample String Assignments

To assign a character string to a local symbol:

```
$ DISK := DKA800  
$ BADGE_NUMBER := 125
```

To assign a character string to a global symbol:

```
$ GLOBAL_DISK ::= DKA800
```

To preserve case and spaces in a string:

```
$ ERROR_MESSAGE_1 := "File NOT found."  
$ GLOBAL_ERROR_MESSAGE ::= "The file was NOT found."
```

Equating Symbols to Integers

Use the equal sign (=) to assign an integer value to a symbol.

- Use one equal sign to create a local symbol:

SYMBOL = INTEGER_VALUE

```
$ COUNT = 1
```

- Use two equal signs to create a global symbol:

SYMBOL == INTEGER_VALUE

```
$ NEWCOUNT == 100
```

Creating a Symbol as a Command Synonym

Equating a symbol to a command line lets you execute the command by typing only the symbol. For example, if you use subdirectories, you may frequently want to change your default directory back to your top level directory.

Create a symbol called HOME that equates to the proper command.

```
$ HOME := SET DEFAULT SYS$LOGIN
```

(or)

```
$ HOME = "SET DEFAULT SYS$LOGIN"
```

If you frequently send print jobs to a printer called TEXT_W, the following symbol will allow you to send a print job by typing TW followed by the file name.

```
$ TW = "PRINT/QUEUE=TEXT_W"
```

To print the file READ_ME.TXT:

```
$ TW READ_ME.TXT
```

To define the symbol UP to set your default to the directory above the current default directory:

```
$ UP = "SET DEFAULT [-]"
```

Creating Symbols that Can Be Abbreviated

Symbols defined with an asterisk (*) may be abbreviated when used. Place the asterisk at the end of the abbreviation. The following example shows how to create a global symbol that can be abbreviated:

```
$ CL*EANUP = "PURGE/LOG"
```

After this symbol is created, any of the following commands will cause the PURGE/LOG command to execute:

```
$ CL
$ CLE
$ CLEA
$ CLEAN
$ CLEANU
$ CLEANUP
```

Displaying Symbols

Sample SHOW SYMBOL Commands

Use the SHOW SYMBOL command to display the equivalence string for a symbol.

SHOW SYMBOL[/GLOBAL][[/ALL] symbol-name

If you want to display...	Use this command:
A local symbol X	\$ SHOW SYMBOL X
All local symbols	\$ SHOW SYMBOL/ALL
A global symbol X	\$ SHOW SYMBOL/GLOBAL X
All global symbols	\$ SHOW SYMBOL/GLOBAL/ALL

Deleting Symbols

DELETE/SYMBOL Command

Use the DELETE/SYMBOL command to cancel a symbol definition.

DELETE/SYMBOL[/GLOBAL] symbol-name

If you want to delete...	Use this command:
A single local symbol	\$ DELETE/SYMBOL X
All local symbols	\$ DELETE/SYMBOL/ALL
A single global symbol	\$ DELETE/SYMBOL/GLOBAL SD
All global symbols	\$ DELETE/SYMBOL/GLOBAL/ALL

Using Keypad Definitions

Overview

You may use key definitions to customize your keyboard so you can enter DCL commands or other strings with fewer keystrokes. A key definition usually contains all or part of a command line. When a key has been defined, you can press that key instead of typing a string of characters.

This section discusses:

- Creating keypad definitions
- Displaying keypad definitions
- Deleting keypad definitions

Creating Keypad Definitions

Keys that Can Be Defined

The type of keyboard you are using determines the keys available for definition. Keypad keys PF1 through PF4 are always enabled for definition. These are on the top row of your numeric keypad, from left to right. The keys F6 through F20 may be defined, if your keyboard has these keys.

Other keys are available if you are willing to give up the use of the numeric keypad on your keyboard. Issue one of the following commands to enable these keys for definition:

```
$ SET TERMINAL/APPLICATION_KEYPAD
$ SET TERMINAL/NUMERIC
```

After issuing either of these commands, you will be able to define the following keys on the numeric keypad:

- KP0 through KP9
- Period (.)
- Comma (,)
- Enter
- Minus (-)

Defining Keys

Use the DEFINE/KEY command to set up keypad definitions at the DCL command level.

DEFINE/KEY key-name equivalence-string[/qualifiers]

Scope and Duration of Keypad Definitions

Keys defined using the DEFINE/KEY command work only at the DCL command level (\$ prompt).

Key definitions remain in effect until you log out or explicitly delete them. If you want key definitions to be available each time you log in, you should put the definitions in your login command procedure, possibly LOGIN.COM.

Tailoring Keypad Definitions

You may tailor a key definition by adding qualifiers to the DEFINE/KEY command.

To cause the key definition to...	Use this qualifier...
Produce an automatic carriage return after the key is entered.	/TERMINATE
Suppress the display of the command being invoked.	/NOECHO
Erase the characters on the current line before displaying and executing the command invoked by the defined key.	/ERASE
Suppress display of an informational message when you initially define a key.	/NOLOG

Example: Enabling and Defining a Key

The following commands enable the numeric keypad for definition and define a key to execute a command.

```
❶ $ SET TERMINAL/NUMERIC
❷ $ DEFINE/KEY COMMA "SHOW TIME"/TERMINATE/NOECHO/NOLOG
❸ $
15-JAN-1999 11:10:43
$
```

Example Notes: Enabling and Defining a Key

- ❶ Enable the keys on the numeric keypad for definition.
The SET TERMINAL/APPLICATION_KEYPAD command does the same thing.
- ❷ Equate the comma key on the numeric keypad to the SHOW TIME command.
- ❸ Press the comma key. The SHOW TIME command is not echoed, but the time is displayed.

Common Errors in Defining Keys

In the following example, the user attempts to define a nonexistent key (KPS):

```
$ DEFINE/KEY KPS "DIRECTORY"
%DCL-W-IVKEYNAM, unrecognized key name - check validity and spelling
$
```

In the following example, the user defines a key, but forgets to use the SET TERMINAL/NUMERIC command:

```
$ DEFINE/KEY KP7 "DIRECTORY"
$ 7 [Return]
%DCL-W-IVVERB, unrecognized command verb - check validity and spelling
\7\
$
```

Displaying Keypad Definitions

Displaying Keypad Definitions

Use the SHOW KEY command to display key definitions.

SHOW KEY [key-name]

Example: Displaying a Specific Keypad Definition

The following command displays the definition for the KP1 key:

```
$ SHOW KEY KP1
DEFAULT keypad definitions:
  KP1 = "SET TERMINAL/NONUMERIC"
$
```

If you do not specify the key, you will be prompted:

```
$ SHOW KEY
_Key name: KP1
DEFAULT keypad definitions:
  KP1 = "SET TERMINAL/NONUMERIC"
$
```

Example: Displaying Detailed Keypad Definitions

Use the SHOW KEY/FULL command to display all qualifiers associated with a key definition.

The following command displays the detailed definition for the COMMA key:

```
$ SHOW KEY/FULL COMMA
DEFAULT keypad definitions:
  COMMA = "RECALL/ALL" (noecho,terminate,noerase,nolock)
$
```

Example: Displaying All Keypad Definitions

Use the SHOW KEY/ALL command to display all of your keypad definitions:

```
$ SHOW KEY/ALL
DEFAULT keypad definitions:
  COMMA = "RECALL/ALL"
  KP1 = "SET TERMINAL/NONUMERIC"
  KP2 = "SHOW KEY/FULL/ALL"
$
```

Deleting Keypad Definitions

Deleting a Single Keypad Definition

Use the DELETE/KEY command to delete a specific key definition.

\$ DELETE/KEY [key-name]

```
$ DELETE/KEY KP4
%DCL-I-DELKEY, DEFAULT key KP4 has been deleted
$
```

Deleting All Keypad Definitions

Use the DELETE/KEY/ALL command to delete all of your key definitions.

\$ DELETE/KEY/ALL

```
$ DELETE/KEY/ALL
%DCL-I-DELKEY, DEFAULT key KP1 has been deleted
%DCL-I-DELKEY, DEFAULT key KP2 has been deleted
%DCL-I-DELKEY, DEFAULT key KP3 has been deleted
$
```

Writing Command Procedures

Overview

Rather than type frequently executed sequences of DCL commands, you may create a file containing these command sequences and direct DCL to execute the file for you. This file, called a *command procedure*, may be executed interactively or in batch mode (to run in the background).

This section discusses:

- Developing command procedures
- Executing a command procedure
- Writing a login command procedure
- SUBMIT command

Developing Command Procedures

Steps for Developing Command Procedures

Follow these steps to develop a command procedure:

1. Design the command procedure.
 - a. Determine what tasks the procedure should perform.
 - b. Decide what results the procedure should produce.
2. Create the command procedure file.
 - a. Use the text editor of your choice.
 - b. Assign the command procedure the file type COM.
3. Execute and test the command procedure.
4. Modify and retest the command procedure, if necessary, by repeating steps 2 and 3.
5. Add comments to simplify maintenance of the command procedure.

Comments might include:

- The name of the procedure and its author
- A description of what the command procedure does, including the kind(s) of input expected
- Descriptions of specific commands and groups of commands included in the command procedure

Rules for Developing Command Procedures

When you write a command procedure, follow these rules:

- Start each DCL command on a new line.
- Precede DCL commands with a dollar sign (\$).
- Use a hyphen (-) to continue a command on the next line (no dollar sign on the continued line).
- Precede comments with an exclamation point (!).

Guidelines for Developing Command Procedures

In addition to the basic rules for constructing a command procedure, consistent format makes your command procedures easy to read, test, and maintain.

- Assign the default file type, COM, to all command procedures. When you want to execute the command procedure you will not need to specify a file type.
- Use full DCL commands, no abbreviations. The full commands will not change in future versions of OpenVMS, but the abbreviated commands may no longer be unique as more commands are added to the language.
- For readability, use blank command lines, containing only the initial \$! characters, to separate blocks of commands.

End your command procedure with the EXIT command so that the end point of your command procedure is clearly indicated. This will help with maintenance.

Example: Command Procedure

Following is an example of a simple command procedure that executes when the owner logs in to the system:

```
$! LOGIN.COM for Dana Stephens
$!
$! Display the date and time on the terminal
$ SHOW TIME
$!
$! Show the default disk and directory
$ SHOW DEFAULT
$!
$! Define some logical names
$! To send mail to all members of my group
$ ASSIGN "SMITH,JONES, LONG, READ" DPD_GROUP
$!
$! Define global symbols
$! Return to login directory
$ HOME == "SET DEFAULT SYS$LOGIN"
$! List file creation dates
$ DA == "DIRECTORY/DATE"
$! Change action of DELETE
$ DEL*ETE == "DELETE/LOG"
$!
$! Define keypad
$ DEFINE/KEY KP4 "RECALL/ALL"/TERMINATE/NOECHO/NOLOG
$!
$ EXIT
```

Executing a Command Procedure

How a Command Procedure Is Executed

DCL command procedures are executed by the DCL Command Language Interpreter. You cannot compile a DCL command procedure.

Command procedures may be executed interactively from the DCL command line.

When a command procedure executes interactively in this manner, output is written to your terminal. Your terminal will not respond to any commands while the procedure is executing.

To execute a command procedure interactively, type the @ symbol followed by the file specification of the procedure.

@command_procedure_name.COM

Example: Executing a Command Procedure

To execute a command procedure called CLEANUP.COM:

\$ @CLEANUP

If no file type is specified, DCL uses the default file type of COM.

Testing Your Command Procedure

Use the SET VERIFY command when you test and debug your command procedures. Add the command as the first line in your command procedure. This will cause the system to print each line of your command procedure before executing the line. Adding the SET NOVERIFY command at the end of the procedure just before the EXIT command will turn this feature off.

When you finish testing your command procedure, remove the SET VERIFY command or make it a comment by adding an exclamation point (!) in front of the command.

You may also enter the SET VERIFY and SET NOVERIFY commands interactively.

Example: Using SET VERIFY

```
❶ $ SET VERIFY
❷      $ @LOGIN

      $! LOGIN.COM for Anne Smith
      $!
      $! Display the date and time on the terminal
❸ $ SHOW TIME

21-JAN-1999 14:30:01

      $! Show the default disk and directory
      $ SHOW DEFAULT
USER8:[SMITH]
      $! Define some logical names
      $! To send mail to all members of my group
      $ ASSIGN "SMITH,JONES, LONG,BAUER"  DPD_GROUP
      $! Define global symbols
      $ HOME == "SET DEFAULT SYS$LOGIN"    ! To return to login directory
      $ DA == "DIRECTORY/DATE"             ! List creation dates of files.
      $ DEL*ETE == "DELETE/LOG"            ! Change action of DELETE command.
      $! Define keypad
      $ DEFINE/KEY KP4 "RECALL/ALL"/TERMINATE/NOECHO/NOLOG
      $ EXIT

❹ $ SET NOVERIFY
      $
```

Example Notes: Using SET VERIFY

- ❶ Enter the SET VERIFY command to cause the system to print each line.
- ❷ Execute a command procedure called LOGIN.COM.
- ❸ Each line of the command procedure is displayed before it is executed.
Any output from a command will be displayed following the command.

After the command procedure exits, you may enter the SET NOVERIFY command to turn off verification.

Halting Execution

You can halt the execution of an interactive command procedure by typing **[Ctrl/Y]** or **[Ctrl/C]**.

Use the `CONTINUE` command to resume execution if you do not execute any intervening DCL commands.

Typing **[Ctrl/C]** may cause some programs to take special action.

Executing a Command Procedure in Batch

You may execute a command procedure as a batch job by using the `SUBMIT` command to place the job in the batch queue.

```
$ SUBMIT CLEANUP
```

Executing a command procedure as a batch job allows you to use your terminal for other work while the command procedure is executing.

Writing a Login Command Procedure

LOGIN.COM

You may create a command procedure to execute the same commands each time you log in to your system. Name the file LOGIN.COM and place it in your top-level directory, unless your system manager tells you otherwise.

The file should contain the commands to set up your OpenVMS environment, including symbols, logical names and key definitions. These will be available to your process each time you log in.



Note

In your LOGIN.COM, you may want to make use of lexicals. OpenVMS lexicals are a set of functions that return information about character strings and attributes of the current process.

For example, F\$MODE returns a character string showing the mode in which a process is executing. The F\$MODE function has no arguments, but must be followed by parentheses.

Bypassing LOGIN.COM

If you modify your LOGIN.COM, be sure to test it before you log out. If your LOGIN.COM contains errors that prevent you from logging in, bypass it by adding the /NOCOMMAND qualifier to your user name when you attempt to log in:

Username: USER1/NOCOMMAND

Example: LOGIN.COM

Copying someone else's and making minor adjustments creates most first login command procedures. As the user becomes more comfortable and skilled in DCL, the login procedure is modified to suit the user's specific needs.

This sample login command procedure illustrates some of the rules and guidelines that we have discussed in this section and also contains some additional advanced information that we have not discussed. It may be used as a template.

```

❶  $! LOGIN.COM                12-JAN-1999   M. Smith
❷  $! Added key definitions     14-JAN-1999   M. Smith
❸  $
    $! Test to see if this is a batch job.
    $! If yes skip to batch commands.
    $
❹  $ IF F$MODE() .EQS. "BATCH" THEN GOTO BATCH_COMMANDS
    $
    $! Do some login work
    $ SHOW TIME
    $
    $! Logical Name Assignments
    $     ASSIGN SYS$LOGIN_DEVICE:[SMITH.WORKSHEET]   WORKSHEET
    $     ASSIGN "BAUER, LONG, SMITH"   DPD_GROUP
    $
    $! Global Symbol Definitions (Available after this procedure
    $ ends.)
❺  $ PW == "PRINT/QUE=TEXT_W "
    $ HOME == "SET DEFAULT SYS$LOGIN:"
    $ PUBLIC == "SET DEFAULT [SMITH.PUBLIC]"
    $
    $! Define keypad
    $
    $     DEFINE/KEY KP4 "RECALL/ALL"/TERMINATE/NOECHO/NOLOG
    $
❻  $ GOTO THE_END              ! End of Interactive commands
    $!
❼  $ BATCH_COMMANDS:
    $     SHOW TIME
    $     THE_END:
❽  $ EXIT

```

Example Notes: LOGIN.COM

- ❶ First comment should be file name, creation date, author
- ❷ Comment to maintain modification history
- ❸ Use lines with dollar sign only to separate the sections of the command procedure for readability.
- ❹ Advanced features: IF-THEN-ELSE, GOTO, and lexical function. This line tests to see if this login is for a batch job, and if it is, skips the commands for interactive processing.
- ❺ Global symbol definitions. You will be able to use these symbols after this command procedure completes processing.
- ❻ GOTO statement ends commands for interactive processing.
- ❼ Commands for batch processing go here.
- ❽ End procedure with EXIT.

Common Errors in Login Command Procedures

The following is a list of common errors with suggested solutions.

- **You cannot log in.**
Check to see if LOGIN.COM contains a LOGOUT command. Log in without allowing LOGIN.COM to execute (/NOCOMMAND), and then edit the LOGOUT command out of your LOGIN.COM file.
- **None of your customizations are in effect.**
These features will not be available to you until LOGIN.COM has executed. Either log out and log back in, or execute LOGIN.COM by typing the following command:

@LOGIN

- **Some symbols do not work.**
Symbols you want to use outside of LOGIN.COM must be global, not local. Did you create these symbols using two equal signs (:=)?

\$ HOME := "SET DEFAULT SYS\$LOGIN"
- **Some key definitions do not work.**
You must execute one of the following commands to enable your keypad for definition:

\$ SET TERMINAL/NUMERIC
\$ SET TERMINAL/APPLICATION_KEYPAD
- **Some DCL commands do not work as they should.**
Have you created a symbol that is the same as a DCL command? The symbol will override the DCL command:

\$ DEL*ETE := "LOGOUT"
- **Some features work and others do not.**
There is an error in your LOGIN.COM file, so some commands were not executed. Enter the SET VERIFY command, then test your LOGIN.COM interactively and watch as each command executes to see where execution stops.
- **You receive "directory not found" error messages.**
You copied someone else's LOGIN.COM and did not replace their directory specification with your own. Edit the LOGIN.COM file and make sure that you have specified the correct directories.

Summary

Concepts

Using Logical Names

- Use the ASSIGN or the DEFINE command to create a logical name.
- Use a logical name in place of all or part of a file specification.
- Use logical names to simplify the process of sending mail to one user or a group of users.
- Use the SHOW LOGICAL or the SHOW TRANSLATION command to determine the equivalence of a logical name.
- The system creates logical names for your process.
- Use the DEASSIGN command to delete a logical name.

Using Symbols

- Create symbols to use as synonyms for frequently used commands.
- Use one or two equal signs (=) to equate a symbol name with a value, depending on whether the symbol is local or global.
- Use the SHOW SYMBOL command to display the contents of a symbol.
- Use the DELETE/SYMBOL command to delete a symbol.

Using Keypad Definitions

- To reduce the amount of typing you have to do, use the DEFINE/KEY command to assign definitions to certain terminal keys.
- Use the SHOW KEY command to display key definitions.
- Use the DELETE/KEY command to delete a key definition.

Writing Command Procedures

- Consistent format makes command procedures easy to read, test, and maintain.
- Use the @ command to execute a command procedure interactively.
- Use the SUBMIT command to execute a command procedure in batch.
- Use your LOGIN.COM command procedure to set up logical names, symbols, keypad definitions, and other characteristics of your terminal session each time you log in.

Commands

Using Logical Names

ASSIGN

Define a logical name.

DEFINE

Define a logical name.

SHOW DEVICE

Display information on a disk or other device.

SHOW LOGICAL

Display a logical name iteratively.

SHOW TRANSLATION

Display a logical name.

DEASSIGN

Remove a logical name.

Using Symbols

SHOW SYMBOL

Display a symbol.

DELETE/SYMBOL

Cancel a symbol definition.

Using Keypad Definitions

DEFINE/KEY

Define a key.

SHOW KEY

Display a key definition.

DELETE/KEY

Remove a key definition.

Writing Command Procedures

@filespec

Execute a command procedure interactively.

SET [NO]VERIFY

Test a command procedure.

SUBMIT filespec

Execute a command procedure in batch mode.