

OpenVMS System and Network Node Management I

Instructor Guide

Training



COMPAQ

OpenVMS System and Network Node Management I

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, BackPaq, COMPAQ, Compaq Insight Manager, CompaqCare logo, Counselor, DECterm, Deskpro, DIGITAL, DIGITAL logo, DIGITAL Alpha Systems, Digital Equipment Corporation, DIGITAL UNIX, DirectPlus, FASTART, Himalaya, InfoPaq, Integrity, LicensePaq, Ministation, NetFlex, NonStop, OpenVMS, PaqFax, Presario, ProLiant, ProLinea, ProSignia, QuickBack, QuickFind, Qvision, RDF, RemotePaq, RomPaq, ServerNet, SERVICenter, SmartQ, SmartStart, SmartStation, SolutionPaq, SpeedPaq, StorageWorks, Systempro/LT, Tandem, TechPaq, 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 System and Network Node Management I

Instructor Guide

December 1999

Overview

Introduction	Overview-1
Topics.....	Overview-1
Course Description	Overview-1
Prerequisites	Overview-2
Resources	Overview-2
Course Organization	Overview-3
Course Goals	Overview-4
Course Nongoes	Overview-5
Course Conventions.....	Overview-6
Materials.....	Overview-7
Introduction.....	Overview-7
Student Package	Overview-7
Instructor Package	Overview-7
Preparing to Teach.....	Overview-9
Introduction.....	Overview-9
If You are New to Teaching OpenVMS System and Network Node Management.....	Overview-9
Preparing the System	Overview-10
Selecting Equipment.....	Overview-10
Creating Student Accounts	Overview-10

Module 1 — Understanding the System Management Environment

About This Module	1-2
Introduction	1-2
Objectives.....	1-2
System Configurations.....	1-3
Overview	1-3
Topics.....	1-3
Single-Processor Configurations	1-4
Overview	1-4
Tightly Coupled Multiprocessor Configurations.....	1-5
Overview	1-5
Tightly Coupled Multiprocessor	1-5
Figure: Tightly Coupled System Configuration.....	1-6
Figure Notes: Tightly Coupled System Configuration.....	1-6
OpenVMS Cluster Systems.....	1-7
Overview	1-7
Characteristics of an OpenVMS Cluster System	1-7

Loosely Coupled Network Configurations	1-8
Overview	1-8
Characteristics of a Network	1-8
Comparison of Multiple Processor Configurations	1-9
OpenVMS Cluster Systems	1-10
Overview	1-10
Topics	1-10
Members of a Cluster	1-11
Special Members of a Cluster	1-11
Boot Servers	1-12
Satellites	1-12
Disk Servers	1-12
Tape Servers	1-12
Ethernet Members	1-13
Interconnects	1-13
CI Bus	1-13
Network Interconnect	1-13
FDDI Network Interconnect	1-13
DSSI Bus	1-14
ATM Connections	1-14
MI Bus	1-14
Sample OpenVMS Cluster Configurations	1-15
OpenVMS Cluster Configurations	1-15
CI OpenVMS Cluster System Configuration	1-16
Figure: CI OpenVMS Cluster System Configuration	1-16
Ethernet OpenVMS Cluster System Configuration	1-18
Figure: Ethernet OpenVMS Cluster System Configuration	1-19
Figure Notes: Ethernet OpenVMS Cluster System Configuration	1-19
FDDI OpenVMS Cluster System Configuration	1-20
Figure: FDDI OpenVMS Cluster System Configuration	1-20
Figure Notes: FDDI OpenVMS Cluster System Configuration	1-21
DSSI OpenVMS Cluster System Configuration	1-22
Figure: DSSI OpenVMS Cluster System Configuration	1-22
Mixed-Interconnect OpenVMS Cluster System Configuration	1-23
Figure: Mixed-Interconnect Configuration	1-24
Figure Notes: Mixed-Interconnect Configuration	1-24
Connection Manager	1-25
Functions of the Connection Manager	1-25
Quorum	1-26
Quorum Disk	1-26
Device Names in an OpenVMS Cluster System	1-27
Naming Devices	1-27
Device Name Formats	1-27
Duplicate Device Names	1-27
Naming Devices on a Local Node	1-28
Figure: Sample OpenVMS Cluster System	1-29
Figure Notes: Sample OpenVMS Cluster System	1-30

Network Configurations.....	1-31
Overview	1-31
Topics.....	1-31
Network Concepts	1-32
Definition of Network.....	1-32
Need for Networks	1-32
Responding to Growing Needs.....	1-33
Nodes	1-34
What the User Can Do Over the Network	1-34
Client/Server Computing	1-35
Figure: Client/Server Computing	1-35
Network Components	1-36
Figure: DECnet Network	1-37
Figure Notes: DECnet Network	1-37
Network Management.....	1-38
Network Management Functions	1-38
Three Levels of Network Management	1-39
Multinode Network	1-39
Area Network	1-39
Multiarea Network.....	1-39
OpenVMS Software Concepts	1-40
Overview	1-40
Programs, Images, and Utilities.....	1-41
Programs	1-41
Object Files	1-41
Images	1-41
Utilities.....	1-42
Processes	1-43
Types of Process.....	1-43
Detached Process.....	1-44
Subprocess.....	1-45
Batch Process	1-45
Network Process.....	1-45
Creating Detached, Batch, and Subprocesses	1-46
Characteristics of a Detached Process	1-46
Characteristics of a Batch Process.....	1-46
Characteristics of a Subprocess.....	1-47
Example: Spawning a Subprocess.....	1-48
Example Notes: Spawning a Subprocess	1-48
System Files, Directories, and Logical Names.....	1-49
OpenVMS Device Names	1-49
Some Common Device Codes.....	1-50
System Files	1-50
System Directories.....	1-51

SYSSYSROOT	1-52
Concealed Root Directories	1-53
Figure: System Directory Tree	1-54
Related System Logical Names.....	1-56
System File and Directory Protection.....	1-57
Summary	1-58
Concepts.....	1-58
System Configurations.....	1-58
OpenVMS Cluster Systems	1-58
Network Configurations.....	1-59
OpenVMS Software Concepts	1-59

Module 2 — Developing Command Procedures

About This Module	2-2
Introduction	2-2
Objectives.....	2-2
Accessing Local and Global Symbol Tables.....	2-3
Overview	2-3
Topics.....	2-3
Local and Global Symbols	2-4
Two Kinds of Symbols	2-4
Local Symbols.....	2-4
Local Symbol Table.....	2-4
Global Symbols	2-4
Global Symbol Table	2-4
Example: Accessing Local and Global Symbol Tables.....	2-5
Example Notes: Accessing Local and Global Symbol Tables.....	2-5
Output from Example	2-6
Terminating a Command Procedure.....	2-7
Using Symbols in Command Lines	2-8
Overview	2-8
Topics.....	2-8
Automatic Symbol Substitution	2-9
Examples of Automatic Symbol Substitution	2-9
Forcing Symbol Substitution.....	2-10
How to Force Symbol Substitution	2-10
When to Force Symbol Substitution.....	2-10
Example: Performing Symbol Substitution	2-11
Example Notes: Performing Symbol Substitution	2-11
Writing Command Procedures That Perform Terminal I/O	2-12
Overview	2-12
Topics.....	2-12
Logical Names Used in Terminal I/O.....	2-13
System-Defined Logical Names.....	2-13
Commonly Used Process-Permanent Logical Names	2-13
Characteristics of Process-Permanent Logical Names	2-14

Using Help for Command Procedures	2-15
Displaying Information at the Terminal.....	2-16
DCL Commands for Displaying Information at the Terminal.....	2-16
WRITE Command	2-16
The Expression Parameter	2-16
Specifying Data Items.....	2-16
Example: Using WRITE SYS\$OUTPUT.....	2-17
Example Notes: Using WRITE SYS\$OUTPUT.....	2-17
Output from Example	2-17
TYPE Command	2-18
Example: Using TYPE SYS\$INPUT	2-18
Example Output.....	2-18
Getting Information from the User	2-19
DCL Commands for Getting Information from the User	2-19
INQUIRE Command	2-19
READ Command.....	2-20
READ/PROMPT Command	2-20
Example: Terminal I/O in a Command Procedure	2-21
Example Notes: Terminal I/O in a Command Procedure	2-22
Using Symbols as Variables and Parameters	2-23
Overview	2-23
Topics.....	2-23
Using Symbols as Variables.....	2-24
Symbols as Variables.....	2-24
Performing Operations with Symbols.....	2-24
Using Symbols as Parameters for Command Procedures.....	2-25
Symbols as Parameters	2-25
Passing Parameters to a Command Procedure	2-25
Coding Without Parameters	2-26
Coding with Parameters.....	2-27
Example: Displaying P1 Through P8	2-28
Example Notes: Displaying P1 Through P8	2-28
Preserving Case and Spacing in Parameters	2-29
Example: Preserving Case and Spacing in Parameters.....	2-29
Example Notes: Preserving Case and Spacing in Parameters.....	2-29
Passing Integer Parameters	2-30
Example: Passing Integer Parameters.....	2-30
Example Notes: Passing Integer Parameters.....	2-30
Controlling Flow of Execution.....	2-31
Overview	2-31
Topics.....	2-31
Flow Control Commands	2-32
DCL Commands	2-32

Testing for Conditions	2-33
Using the IF-THEN Command	2-33
What Is a Conditional Expression?	2-33
When Is a Conditional Expression True?	2-33
Operators Used in Conditional Expressions	2-34
Comparing Numeric Expressions	2-34
Comparing Strings	2-34
Logical Operators	2-34
Precedence of Operation	2-35
Simple IF Command	2-36
Example: IF Command	2-36
Complex IF Command	2-36
Example: Complex IF Command	2-37
IF THEN ELSE Command	2-38
Example: Using IF-THEN-ELSE	2-38
Evaluation of Expressions in IF Commands	2-39
Transferring Control Within a Command Procedure	2-40
Flow of Execution	2-40
GOTO Command	2-40
Example: GOTO Command	2-40
Labels	2-40
Nonexistent Label	2-41
Duplicate Labels	2-41
Caution About Labels	2-41
Example: Using the GOTO Command	2-42
Example Notes: Using the GOTO Command	2-42
Example: Branching in a Command Procedure	2-43
Example Notes: Branching in a Command Procedure	2-43
Transferring Control Outside of a Command Procedure	2-44
Nesting Command Procedures	2-44
@ Command	2-44
Availability of Symbols at Different Command Levels	2-44
Iterative Command Procedures	2-45
Iteration	2-45
Example: Iterative Command Procedure	2-45
Example Notes: Iterative Command Procedure	2-45
Redirecting Input and Output	2-46
Overview	2-46
Topics	2-46
Redefining SYS\$OUTPUT	2-47
Redefining SYS\$OUTPUT	2-47
Directing Output to a File	2-47
Example: Directing Output to a File	2-48

Example Notes: Directing Output to a File.....	2-48
Temporarily Redefining SYS\$OUTPUT.....	2-49
Example: Temporarily Directing Output to a File	2-49
Example Notes: Temporarily Directing Output to a File.....	2-49
Redefining SYS\$INPUT.....	2-50
Editing a File from a Command Procedure	2-50
Running an Interactive Program from a Command Procedure.....	2-50
Using Lexical Functions	2-51
Overview	2-51
Topics.....	2-51
Lexical Functions	2-52
What Are Lexical Functions?.....	2-52
Format of Lexical Functions	2-52
Arguments.....	2-52
Manipulating Character Strings.....	2-53
F\$EDIT	2-54
Example: F\$EDIT	2-54
F\$LENGTH.....	2-55
Example: F\$LENGTH.....	2-55
F\$LOCATE.....	2-56
Example: F\$LOCATE	2-56
F\$EXTRACT	2-57
Example: F\$EXTRACT.....	2-57
Obtaining and Manipulating Data	2-58
F\$DIRECTORY	2-58
Example: F\$DIRECTORY	2-58
F\$MODE.....	2-59
Example: F\$MODE	2-59
F\$TIME.....	2-60
Example: F\$TIME	2-60
F\$CVTIME	2-61
Example: F\$CVTIME.....	2-62
F\$GETSYI	2-63
Example: F\$GETSYI.....	2-63
Writing Command Procedures That Perform I/O	2-64
Overview	2-64
Topics.....	2-64
File Access Steps.....	2-65
Opening a File	2-66
OPEN Command	2-66
OPEN/READ Command.....	2-67
Example: OPEN/READ.....	2-67
OPEN/WRITE Command.....	2-67
Example: OPEN/WRITE	2-67

OPEN/WRITE /READ Command	2-67
Example: OPEN/WRITE/READ.....	2-67
OPEN/APPEND Command	2-68
Example: OPEN/APPEND	2-68
OPEN/ERROR Command	2-68
Example: OPEN/ERROR	2-68
Writing to a File.....	2-69
WRITE Command	2-69
WRITE/ERROR Command	2-69
WRITE/UPDATE Command.....	2-69
Example: Using WRITE in a Command Procedure.....	2-70
Example Notes: Using WRITE in a Command Procedure.....	2-70
Reading from a File	2-71
READ Command.....	2-71
READ/ERROR Command.....	2-71
READ /END_OF_FILE Command.....	2-72
Example: Using a Command Procedure to Read a File.....	2-72
Example Notes: Using a Command Procedure to Read a File.....	2-72
Closing a File.....	2-73
CLOSE Command.....	2-73
Example: CLOSE	2-73
CLOSE/ERROR Command	2-73
Example: Using OPEN and CLOSE	2-74
Example Notes: Using OPEN and CLOSE.....	2-74
Handling File I/O Errors	2-75
/ERROR Qualifier	2-75
READ /END_OF_FILE Command.....	2-75
Example: Handling File I/O Errors	2-76
Example Notes: Handling File I/O Errors	2-76
Error Conditions and Ctrl/Y	2-77
Overview	2-77
Topics.....	2-77
Detecting Errors in Command Procedures.....	2-78
Condition Code.....	2-78
How Condition Codes Are Returned.....	2-78
Format of \$STATUS	2-79
Values for \$SEVERITY	2-79
Handling Errors in Command Procedures	2-80
Handling Errors.....	2-80
ON Command.....	2-80
Example: ON.....	2-80
Default ON Action.....	2-80
Guidelines for Using the ON Command.....	2-81
Example: Using the ON Command	2-82
Example Notes: Using the ON Command	2-82

SET ON Command.....	2-83
SET NOON Command	2-83
Guidelines for Using the SET NOON Command	2-83
When to Use SET NOON	2-84
Using \$STATUS and \$SEVERITY in Checking for Errors.....	2-84
Example: Using \$SEVERITY to Check for Errors.....	2-85
Example Notes: Using \$SEVERITY to Check for Errors.....	2-85
Handling Ctrl/Y Interrupts	2-86
Ctrl/Y Interrupts	2-86
Trapping Ctrl/Y Interrupts	2-86
Action of the ON CONTROL_Y Command	2-86
Disabling Ctrl/Y Interrupts	2-87
Cautions About Disabling Ctrl/Y Interrupts	2-87
Enabling Ctrl/Y Interrupts	2-87
Summary	2-88
Concepts.....	2-88
Accessing Local and Global Symbol Tables	2-88
Using Symbols in Command Lines	2-88
Writing Command Procedures That Perform Terminal I/O	2-88
Using Symbols as Variables and Parameters.....	2-88
Controlling Flow of Execution.....	2-89
Redirecting Input and Output.....	2-89
Using Lexical Functions	2-89
Writing Command Procedures That Perform File I/O	2-90
Handling Error Conditions and Ctrl/Y	2-90
Commands.....	2-91
Accessing Local and Global Symbol Tables	2-91
Using Symbols in Command Lines	2-91
Writing Command Procedures That Perform Terminal I/O	2-91
Using Symbols as Variables and Parameters.....	2-91
Controlling Flow of Execution.....	2-92
Redirecting Input and Output.....	2-92
Writing Command Procedures That Perform File I/O	2-93
Handling Error Conditions and Ctrl/Y.....	2-93

Module 3 — Managing System Users

About This Module.....	3-2
Introduction	3-2
Objectives.....	3-2
Overview	3-3
Topics.....	3-3
User Authorization File.....	3-4
Running the Authorize Utility.....	3-5
Running the Authorize Utility.....	3-5
Example: Running the Authorize Utility	3-5

Getting Help from the Authorize Utility.....	3-6
Example: Getting Help from the Authorize Utility.....	3-6
Displaying a UAF Record.....	3-7
Displaying a UAF Record.....	3-7
Example: Displaying a UAF Record.....	3-7
Example: Displaying a UAF Record.....	3-8
Example Notes: Displaying a UAF Record.....	3-9
Using the Authorize Utility to List User Accounts	3-10
Overview	3-10
Topics.....	3-10
Using the Authorize Utility to List User Accounts	3-11
Listing User Accounts	3-11
Sample Commands for Listing User Accounts.....	3-11
Information Contained in a Brief UAF Listing.....	3-11
Example: Brief UAF Listing.....	3-12
Example Notes: Brief UAF Listing.....	3-12
Adding a User to the System.....	3-13
Overview	3-13
Topics.....	3-13
Planning the User's Account	3-14
Planning the Account.....	3-14
Assigning a User Name	3-14
Assigning a Password.....	3-15
Minimum Password Length	3-15
Assigning a User Identification Code.....	3-16
Assigning a Default Device	3-16
Assigning a Default Directory.....	3-16
Assigning an Account Name.....	3-16
Adding the UAF Record	3-17
Specifying the Account When Adding a User	3-17
Enabling New User Account.....	3-17
Default UAF Record.....	3-17
Example: Manually Adding the User Record.....	3-18
Example Notes: Manually Adding the User Record	3-18
Establishing a Disk Quota.....	3-19
Disk Quota	3-19
Using SYSMAN to Set Up a Disk Quota.....	3-19
Example: Using SYSMAN to Set Up a Disk Quota	3-20
Example: Default Disk Quota Values.....	3-20
Overbooking a Disk.....	3-20
Creating a Default Directory	3-21
Default Directory.....	3-21
Setting Up a Default Directory.....	3-21

Testing the New Account.....	3-22
Example: Testing a New Account.....	3-22
Example Notes: Testing a New Account	3-23
Resetting the User's Password.....	3-23
Example: Resetting the User's Password	3-24
Example Notes: Resetting the User's Password	3-24
Using an Automated Procedure to Add a User	3-25
ADDUSER.COM	3-25
Example: ADDUSER Session.....	3-25
ADDUSER Session (continued)	3-26
Example Notes: ADDUSER Session.....	3-27
Using OpenVMS Management Tools for Windows NT to Add a	
User Account.....	3-28
What Is OpenVMS Management Station?.....	3-28
Account Management	3-28
Common Errors in Adding a User	3-30
Common Errors	3-30
Error: Not Having Sufficient Privilege to Run AUTHORIZE.....	3-30
Correcting: Not Having Sufficient Privilege to Run AUTHORIZE	3-30
Error: Not Running AUTHORIZE Correctly.....	3-31
Correcting: Not Running AUTHORIZE Correctly	3-31
Error: Assigning a Duplicate UIC	3-32
Correcting: Assigning a Duplicate UIC.....	3-32
Error: Not Specifying Account for a New UIC Group.....	3-33
Correcting: Not Specifying Account for a New UIC Group	3-33
Error: Not Enabling the User's Account	3-34
Correcting: Not Enabling the User's Account.....	3-34
Error: Not Specifying a Default Device/Directory.....	3-34
Correcting: Not Specifying a Default Device/Directory	3-34
Error: Not Specifying Owner of Directory	3-35
Correcting: Not Specifying Owner of Directory.....	3-35
Error: Incorrectly Locating the User's Directory.....	3-36
Correcting: Incorrectly Locating the User's Directory	3-36
Error: Not Setting Up a Disk Quota	3-37
Correcting: Not Setting Up a Disk Quota.....	3-37
Removing a User Account	3-38
Overview	3-38
Topics.....	3-38
Preventing the User from Logging In.....	3-39
Preventing Logins.....	3-39
Preventing the User from Logging In.....	3-39
Removing Disk Files from the User Directories	3-40
Removing User Files	3-40
Deciding What to Do with the User's Files.....	3-40
Steps for Removing a User's Files.....	3-40
Example: Removing a User's Files.....	3-41
Example Notes: Removing a User's Files.....	3-42

Removing Entries from the Quota File	3-43
Removing Quota Entries	3-43
Example: Removing Entries from the Quota File	3-43
Removing Mail Information	3-44
Example: Removing Mail Information	3-44
Removing the UAF Record	3-45
Example: Removing the UAF Record	3-45
Managing and Monitoring User Processes	3-46
Overview	3-46
Topics	3-46
Setting User Limits and Privileges	3-47
User Limits and Privileges	3-47
Granting or Removing Privileges	3-47
Example: Granting and Removing Privileges	3-48
Example Notes: Granting and Removing Privileges	3-48
Establishing Limits	3-49
Example: Establishing Limits	3-49
Specifying Login Command Procedures	3-50
Login Command Procedures	3-50
System Login Command Procedure	3-50
User's Login Command Procedure	3-50
Example: Specifying a Login Command Procedure	3-51
Example Notes: Specifying a Login Command Procedure	3-51
Controlling Processes	3-52
Restricting Processes	3-52
Privileges Required to Control Processes	3-52
Suspending a Process	3-53
Example: Suspending a Process	3-54
Example Notes: Suspending a Process	3-54
Resuming a Process	3-55
Example: Resuming a Process	3-56
Example Notes: Resuming a Process	3-56
Deleting a Process	3-57
Example: Deleting a Process	3-58
Example Notes: Deleting a Process	3-58
Using the SYSMAN Utility to Manage Processes on Other Nodes	3-59
SYSMAN Utility	3-59
Running the SYSMAN Utility	3-59
Establishing a SYSMAN Environment	3-60
Establishing a SYSMAN Profile	3-61
Executing DCL Commands in the Current Environment	3-61
Example: Executing Commands on Other Nodes	3-62
Example Notes: Executing Commands on Other Nodes	3-63

Managing the Use of Disk Space	3-64
Overview	3-64
Topics.....	3-64
Purging Old Versions of Files	3-65
Old Versions of Files	3-65
System Files to Purge	3-65
Setting a Version Limit on Directories	3-66
Version Limits on Directories	3-66
Setting a Version Limit When a Directory Is Created.....	3-66
Example: Setting a Version Limit When a Directory Is Created.....	3-67
Example Notes: Setting a Version Limit When a Directory Is Created....	3-67
Setting a Version Limit on an Existing Directory.....	3-68
Example: Setting a Version Limit on an Existing Directory	3-69
Example Notes: Setting a Version Limit on an Existing Directory	3-69
Monitoring Disk Space	3-70
Example: Displaying Information About a Specific Disk	3-70
Example: Displaying Information About a Group of Disks	3-70
Actions to Take When Disk Space is Low	3-71
Educating Users About Disk Usage	3-71
Managing Disk Quotas	3-72
Creating and Modifying a Disk Quota File.....	3-72
Default Disk Quotas	3-73
Example: Controlling Disk Quotas	3-73
Example Notes: Controlling Disk Quotas.....	3-73
Summary	3-74
Concepts.....	3-74
Using the Authorize Utility.....	3-74
Using the Authorize Utility to List User Accounts	3-74
Adding a User to the System.....	3-74
Using OpenVMS Management Tools for Windows NT for Adding User Accounts	3-75
Removing a User Account	3-75
Managing and Monitoring User Processes	3-75
Managing the Use of Disk Space	3-76
Commands.....	3-77
Using the Authorize Utility.....	3-77
Using the Authorize Utility to List User Accounts	3-77
Adding a User to the System.....	3-77
Removing a User Account	3-78
Managing and Monitoring User Processes	3-78
Managing the Use of Disk Space	3-79

Module 4 — Maintaining Queues

About This Module	4-2
Introduction	4-2
Objectives	4-2
OpenVMS Queues	4-3
Overview	4-3
Topics	4-3
Generic and Execution Queues	4-4
Two Classes of Queues	4-4
Batch and Output Queues	4-5
Queue Manager	4-6
Additional Queue Managers	4-6
Priority of Jobs in a Queue	4-8
Priority	4-8
Example: Priority of a Job in a Queue	4-9
Example Notes: Priority of a Job in a Queue	4-9
Monitoring Queues and Jobs	4-10
Overview	4-10
Topics	4-10
Monitoring Queue Status	4-11
SHOW QUEUE Command	4-11
Example: Output from the SHOW QUEUE Command	4-12
Example Notes: Output from the SHOW QUEUE Command	4-12
Tailoring the SHOW QUEUE Command	4-13
SHOW QUEUE Command	4-13
SHOW QUEUE Qualifiers	4-13
Example: Displaying Generic Queues	4-13
Using the /FULL and /ALL	4-13
Using Wildcards to Display Queue Information	4-14
Identifying SHOW QUEUE Status Codes	4-15
Monitoring Job Status	4-16
SHOW ENTRY Command	4-16
Specifying Job Information to Display	4-16
Example: Displaying Job Status Information	4-16
Job Status Codes	4-17
Stopping a Queue	4-18
Overview	4-18
Topics	4-18
Stopping a Queue	4-19
Pausing a Queue	4-20
STOP/QUEUE Command	4-20
Example: Pausing a Queue	4-21
Example Notes: Pausing a Queue	4-21

Orderly Shutdown of a Queue.....	4-22
Example: Orderly Shutdown of a Queue	4-22
Example Notes: Orderly Shutdown of a Queue	4-22
Immediate Shutdown of a Queue	4-23
Example: Immediate Shutdown of a Queue	4-23
Example Notes: Immediate Shutdown of a Queue.....	4-23
Closing a Queue	4-24
Example: Closing a Queue.....	4-25
Example Notes: Closing a Queue.....	4-25
Shutting Down All Queues on a Single Node.....	4-26
Example: Shutting Down All Queues on a Single Node	4-26
Example Notes: Shutting Down All Queues on a Single Node	4-26
Shutting Down Queues on All Nodes in an OpenVMS Cluster	4-27
Using /NAME_OF_MANAGER	4-27
Example: Shutting Down Queues on All Nodes in an OpenVMS Cluster	4-27
Starting Queues	4-28
Overview	4-28
Topics.....	4-28
Starting the Queue Manager.....	4-29
When to Start the Queue Manager	4-29
Specifying Nodes on Which to Run the Queue Manager	4-30
Starting Additional Queue Managers	4-31
When to Start the Queue Manager	4-31
Specifying Nodes on Which to Run the Queue Manager	4-32
Starting a Queue	4-33
Example: Starting a Queue.....	4-33
Example Notes: Starting a Queue.....	4-34
Changing the Characteristics of a Queue.....	4-35
Overview	4-35
Topics.....	4-35
Changing Queue Characteristics	4-36
Sample SET QUEUE Qualifiers	4-36
Characteristics of Queues.....	4-36
Limiting Job Sizes	4-37
Job Size	4-37
Limiting Job Sizes	4-37
Scheduling Print Jobs by Size	4-38
Example: Changing Queue Characteristics.....	4-38
Example Notes: Changing Queue Characteristics.....	4-39
Managing Jobs in a Queue	4-40
Overview	4-40
Topics.....	4-40
Aborting and Requeuing Jobs on Output Queues	4-41
Aborting and Requeuing Jobs	4-41
Example: Requeuing a Print Job	4-42
Example Notes: Requeuing a Print Job	4-42

Moving a Job to a Different Queue	4-43
Example: Moving a Job to a Different Queue	4-44
Example Notes: Moving a Job to a Different Queue.....	4-44
Merging Queues	4-45
Example: Merging Queues.....	4-45
Example Notes: Merging Queues.....	4-45
Changing the Characteristics of a Job.....	4-46
Changing Job Characteristics	4-46
Example: Changing the Characteristics of a Job.....	4-46
Example Notes: Changing the Characteristics of a Job.....	4-46
OpenVMS Management Tools for Windows NT (OMT)	4-47
Printer Management	4-47
Summary	4-48
Concepts.....	4-48
OpenVMS Queues.....	4-48
Monitoring Queues and Jobs.....	4-49
Stopping a Queue	4-49
Starting Queues	4-50
Changing the Characteristics of a Queue.....	4-50
Managing Jobs in a Queue	4-50
Uses for OpenVMS Management Tools for Windows.....	4-51
Commands.....	4-52
Monitoring Queues and Jobs.....	4-52
Stopping a Queue	4-52
Starting Queues	4-52
Changing the Characteristics of a Queue.....	4-53
Managing Jobs in a Queue	4-53

Module 5 — Performing Backups and Restores

About This Module.....	5-2
Introduction	5-2
Objectives.....	5-2
Overview	5-3
Topics.....	5-3
Backup Terminology	5-4
Types of Backups	5-4
Save Sets	5-5
Planning Backups	5-6
Scheduling Backups.....	5-6
Backup Media.....	5-6
Open Files During a Backup	5-7
Informing Users About Backups.....	5-7
BACKUP Command.....	5-8
Parts of a BACKUP Command.....	5-8
Input and Output Specifications	5-8
Using Wildcards in a BACKUP Command.....	5-9

Making Backups of a Disk to Tape	5-10
Overview	5-10
Topics.....	5-10
Preparing Disks and Tapes for Backups	5-11
Preparing Volumes for Backups.....	5-11
Preparing Tapes for Backups	5-11
How BACKUP Mounts Devices	5-11
Volume Labels.....	5-12
Making a Full Backup.....	5-13
Full Backup	5-13
Making a Full Backup.....	5-13
/RECORD Qualifier.....	5-14
/EXACT_ORDER Qualifier	5-14
Example: Full Backup of Disk to Tape	5-15
Example Notes: Full Backup of Disk to Tape	5-15
Multivolume Backups.....	5-16
Volume Labels.....	5-16
Making an Incremental Backup.....	5-17
Incremental Backup	5-17
Making an Incremental Backup	5-17
Example: Performing an Incremental Backup	5-17
Example Notes: Performing an Incremental Backup	5-18
Using Command Procedures for Backups	5-19
Overview	5-19
Topics.....	5-19
Nightly Image Backup	5-20
Example: Command Procedure for Nightly Image Backups.....	5-20
Example Notes: Command Procedure for Nightly Image Backups.....	5-20
Nightly Incremental Backup	5-21
Example: Command Procedure for Nightly Incremental Backup.....	5-21
Example Notes: Command Procedure for Nightly Incremental Backup...	5-21
Restoring Files from Backup Copies.....	5-22
Overview	5-22
Topics.....	5-22
Using the BACKUP Command to Restore Files.....	5-23
BACKUP Restore Operation	5-23
BACKUP Command	5-23
Restoring All Files on a Disk	5-24
Restoring an Entire Disk.....	5-24
Procedure for Restoring an Entire Disk.....	5-24
Initializing the Output Disk.....	5-25
Example: Restoring a Disk from a Full Backup.....	5-26
Example Notes: Restoring a Disk from a Full Backup.....	5-26

Restoring Files from an Incremental Backup.....	5-27
Order for Restoring Files	5-27
Procedure for Restoring Files from an Incremental Backup.....	5-28
Example: Restoring a Series of Incremental Backups.....	5-29
Example Notes: Restoring a Series of Incremental Backups.....	5-29
Restoring Specific Files	5-30
Example: Restoring Specific Files	5-30
Restoring a Directory Tree.....	5-31
Example: Restoring a Directory Tree	5-31
Listing the Contents of a Save Set.....	5-32
Overview	5-32
Topics.....	5-32
Using BACKUP to List Contents of Save Sets.....	5-33
Listing a Save Set	5-33
Example: Listing a Save Set	5-34
Example Notes: Listing a Save Set	5-34
Summary	5-35
Concepts.....	5-35
Backup Concepts.....	5-35
Making Backups of a Disk to Tape	5-35
Using Command Procedures for Backups	5-36
Restoring Files from Backup Copies.....	5-36
Listing the Contents of a Save Set.....	5-36
Commands.....	5-37
Backup Concepts.....	5-37
Making Backups of a Disk to Tape	5-37
Restoring Files from Backup Copies.....	5-37
Listing the Contents of a Save Set.....	5-37

Module 6 — Shutting Down and Starting Up an OpenVMS System

About This Module.....	6-2
Objectives.....	6-2
Shutting Down an OpenVMS System	6-3
Overview	6-3
Topics.....	6-4
Orderly Shutdown Using SHUTDOWN.COM.....	6-5
What Happens in an Orderly Shutdown	6-5
Procedure to follow for Orderly Shutdown.....	6-6
Interrupting the Shutdown	6-6
Functions of SHUTDOWN.COM.....	6-7
SHUTDOWN.COM Options	6-8
Notification of OpenVMS Cluster Members	6-8
Customizing SHUTDOWN.COM.....	6-8
Example: Orderly Shutdown.....	6-9
Example: Orderly Shutdown (continued)	6-10

Example Notes: Orderly Shutdown.....	6-11
Fast Orderly Shutdown	6-11
Shutdown Command Files	6-11
Emergency Shutdown	6-12
Emergency Conditions.....	6-12
Procedure to Follow for Emergency Shutdown	6-12
Example: Emergency Shutdown	6-13
Example Notes: Emergency Shutdown	6-13
Emergency Shutdown Without DCL.....	6-14
Starting Up and OpenVMS System.....	6-15
Overview	6-15
Topics.....	6-15
Startup Command Procedures.....	6-16
Command Procedures.....	6-16
Functions of STARTUP.COM.....	6-17
Functions of SYSTARTUP_VMS.COM.....	6-18
Booting Satellite Nodes	6-19
Overview	6-19
Topics.....	6-19
Checking the Status of Boot Service	6-21
Status of Boot Service.....	6-21
Example: Using NCP to Check Boot Service Status.....	6-21
Example Notes: Using NCP to Check Boot Service Status.....	6-22
Enabling Boot Service	6-23
Boot Service	6-23
Example: Enabling Satellite Boot Service	6-23
Example Notes: Enabling Satellite Boot Service	6-24
Disabling Boot Service	6-25
Boot Service	6-25
Example: Disabling Satellite Boot Service.....	6-25
Example Notes: Disabling Satellite Boot Service	6-26
SERVICE_ON.COM and SERVICE_OFF.COM.....	6-27
Command Procedures.....	6-27
Example: SERVICE_ON.COM.....	6-27
Example: SERVICE_OFF.COM.....	6-27

Summary	6-28
Concepts.....	6-28
Shutting Down an OpenVMS System	6-28
Starting Up an OpenVMS System.....	6-28
Booting Satellite Nodes	6-28
Commands.....	6-29
Shutting Down an Open VMS System	6-29
Booting Satellite Nodes	6-29

Learning Check

Learning Check Answers

Glossary

Introduction

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

Topics

This section discusses the following topics:

- Course description
- Prerequisites
- Resources
- Course organization
- Course goals
- Course nongoes
- Course map
- Course conventions

Course Description

OpenVMS System and Network Node Management I is designed to provide introductory training for the system manager of an established VAX or Alpha computer or OpenVMS Cluster system running the OpenVMS operating system. This system may be part of a larger network or cluster.

Prerequisites

Students taking this course 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 an 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 operating 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

These skills can be obtained by taking the *OpenVMS Skills for Users* course, or through equivalent experience on an OpenVMS system.

Resources

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

- *DECams User's Guide*
- *Digital OpenVMS System Messages: Companion Guide for Help Message Users*
- *Guide to the DEC Text Processing Utility*
- *Guidelines for OpenVMS Cluster Configurations*
- *OpenVMS Alpha Version 7.2 Upgrade and Installation Manual*
- *OpenVMS Alpha Galaxy Guide*
- *OpenVMS Guide to System Security*
- *OpenVMS Cluster Systems*
- *OpenVMS DCL Dictionary*
- *OpenVMS License Management Utility Manual*
- *OpenVMS Management Station Overview and Release Notes*
- *OpenVMS System Management Utilities Reference Manual*
- *OpenVMS System Manager's Manual*
- *OpenVMS User's Manual*

- *OpenVMS VAX Version 7.2 Upgrade and Installation Manual*
- *OpenVMS Version 7.2 New Features Manual*
- *OpenVMS Version 7.2 Release Notes*
- *Overview of OpenVMS Documentation*
- *TCP/IP Networking on OpenVMS Systems*
- *VAX MACRO and Instruction Set Reference Manual S Volume Shadowing for OpenVMS*

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.

The **glossary** contains definitions for many terms used in this course.

Course Goals

To perform day-to-day system management tasks on an established system, a system and network node manager should be able to:

- Demonstrate an understanding of the system management environment
 - Describe basic OpenVMS system terminology
 - Explain the basic concepts of a network
 - Explain the basic concepts of a cluster
 - Identify directories on a system disk
 - Classify OpenVMS processes
- Develop command procedures that:
 - Use terminal I/O
 - Redirect input and output to a file
 - Use lexical functions
 - Use local and global symbols
 - Control the flow of execution
 - Detect and handle error conditions
- Manage system users
 - Use the Authorize utility to manipulate user records
 - Add and remove system users
 - Manage the use of disk space
 - Manage, monitor, and control user processes
- Maintain queues
 - Identify the different types of queues
 - Start a queue
 - Stop a queue
 - Manage and monitor queue usage

- Back up files and disks
 - Identify reasons to back up files and disks
 - Explain backups to users
 - Back up a set of files
 - List the contents of a save set
 - Restore files from a save set
 - Back up a disk volume
 - Restore a disk volume
- Shut down and start up an OpenVMS system
 - Explain why system shutdowns and startups are used
 - Shut down a system
 - Explain the differences between CCL and DCL
 - Start up a system
 - Examine network status

Course Nongoals

This course does not cover the following topics:

- System installation (including OpenVMS operating system installation), adding nodes to an OpenVMS Cluster system, or adding nodes to a network
- System customization, including customizing boot procedures and creating queues
- System performance management and tuning
- System security features other than file protection
- System troubleshooting
- Programming using system services, run-time libraries, or other OpenVMS features
- Advanced OpenVMS concepts or system programming
- Details of layered product features and functions

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, ...	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 [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.)

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 one week. Students receive a course guide, including both lecture and lab materials, and several manuals for supplementary reading and reference.

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

Student Package

Each student in the class receives a student package containing:

- Course Guide
- Exercises

Instructor Package

Each instructor should receive an Instructor Package containing:

- This Instructor Guide

In addition, provide a full OpenVMS document set in the lab or classroom for reference.

The course guide is designed as an aid for the student during lecture. As you explain a particular subject, you can refer the student to the corresponding outline in the guide, and the student can take notes in the blank areas provided. By using the topic outlines, tables, figures, and examples as a framework, the student will be able to channel his or her energies into understanding the subject matter and away from recording it.

Transparencies

Many instructors like to illustrate their lectures with transparencies duplicated from the pages of the course guide. To keep this textbook to a manageable size, the lettering in the outlines is not large, and the lettering in the tables and examples is smaller still. Your transparencies will be more effective if you have them typed using large lettering.

Using Tables

Many of the system manager's functions are explained using tables. Most tables are organized according to the function to be performed, not the utility or command used. It is much more interesting to learn how to perform a sequence of useful jobs than it is to study an alphabetical sequence of commands and qualifiers.

The tables usually contain command formats, examples, and explanatory comments. They are intended for use in the lab and on the job as well as in class. When you lecture, you are not expected to teach every function in every table, nor to restrict yourself to what is in the tables. By all means, explain the material in your own way. But refer to the tables as you go along, so that the student will be familiar with them when he or she is in the lab and on the job.

Hardware Needed

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 available for every two students.

Exercises

Every module provides simple exercises that allow the student to practice basic concepts and commands. Several modules provide more advanced exercises to challenge your more prepared students.

Preparing to Teach

Introduction

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

- An instructor package
- A copy of each document listed under "Additional References"
- The OpenVMS document set

If You are New to Teaching OpenVMS System and Network Node Management

If you are not familiar with the material, take the self-paced courses, *OpenVMS System and Network Node Management I, II, and III*. It would also be helpful to sit in on *OpenVMS System and Network Node Management* lecture lab classes taught by an experienced instructor. If you have not taken *VMS System Architecture*, you should take that course as well.

In addition, after you learn the subject matter of this course, you should attend the *OpenVMS System Performance Management* course, or, if this is not possible, the *VMS Internals I and II* courses. Familiarity with (not necessarily complete mastery of) the subject matter of these courses will give you a better perspective on the *OpenVMS System and Network Node Management* courses.

You should read each laboratory exercise and be sure you have the materials available to support them. A later section lists the type of preparation required for each lab.

There is no substitute for experience. Building and managing an OpenVMS system at your training center will give you valuable background for teaching this course.

The following schedule can be used as a guide:

Day	Modules	Comment
1	1	This module describes the system management environment, from hardware and software to typical tasks.
2	2	As you lecture, build a command procedure that builds on what the students are learning and increases in complexity.
3	3	
4	4–5	
5	6	

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 the OpenVMS operating system
- A terminal for each student

The exercises for this course call for installing, starting up, and reconfiguring the OpenVMS system in many ways. These activities interfere with students using the system for other classes. Furthermore, students with the privileges necessary to perform these exercises may unintentionally affect system behavior in many undesirable ways. For these reasons, it is strongly recommended that the labs be run on a standalone machine with disk volumes dedicated exclusively to courses in which students have unlimited privileges.

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.

The student accounts should have the following privileges, at minimum:

- TMPMBX — allows use of queues
- NETMBX — allows use of DECnet software

Other privileges are needed to perform some operations illustrated in the course. Those privileges are potentially dangerous, and you may want to restrict their use. Some operations also require physical access to the system hardware and need to be done under your supervision.

If the course is given on a standalone system, allow each student to have the SETPRV privilege and allow students to give themselves the privileges they need to perform various operations.

Understanding the System Management Environment

Module 1

About This Module

Introduction

System and network managers have many roles and responsibilities. They must handle issues surrounding hardware configurations and they must also deal with software compatibility and requirements. System Managers must balance both hardware and software requirements with the needs of the users and the business.

Although sites vary in their installed equipment and configuration, there are many common elements to all sites.

Objectives

To describe the OpenVMS system and network environment, you should be able to:

- Identify various OpenVMS system configurations.
- Describe the characteristics of an OpenVMS Cluster system.
- Describe the characteristics of a network and list basic network management tasks.
- Describe OpenVMS software concepts.

System Configurations

Overview

You can build many different configurations with the various Alpha and VAX processors, interconnects, and peripheral devices described in this section.

Topics

In this section the following topics are discussed:

- Single-processor configurations
- Tightly coupled multiprocessor configurations
- OpenVMS Cluster systems
- Loosely coupled configurations (networks)
- Comparison of multiple processor configurations



Note

This material is intended to be a general overview of the hardware and software pieces that make up the OpenVMS environment. Although the emphasis is on clusters and networks, be aware that many sites are not clustered; they may have workstations for each person that are managed by the owner, or there may be a small system with many terminals attached.

Single-Processor Configurations

Overview

A single-processor configuration is a single independent processor and its peripheral devices. This could be a single workstation sitting on a desk, managed by the owner, or it could have many terminals attached, supporting many users and managed by one person.



Note

This material is intended to be a general overview of the hardware and software pieces that make up the OpenVMS environment. Although the emphasis is on clusters and networks, be aware that many sites are not clustered; they may have workstations for each person that are managed by the owner, or there may be a small system with many terminals attached.

Tightly Coupled Multiprocessor Configurations

Overview

In a tightly coupled configuration:

- CPUs share operating system code
- CPUs cannot operate independently from each other
- Management and operation are handled as a single machine
- The environment provides high performance

Tightly Coupled Multiprocessor

In a tightly coupled multiprocessor configuration:

- Two or more CPUs share memory by means of a high speed bus.
- A multiprocessor configuration acts as a single system to users and to system managers.



Note

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

A tightly coupled system is operated and managed as a single domain. The attached processor is transparent to the system users. To the user, a tightly coupled system appears the same as a single processor configuration.

Figure: Tightly Coupled System Configuration

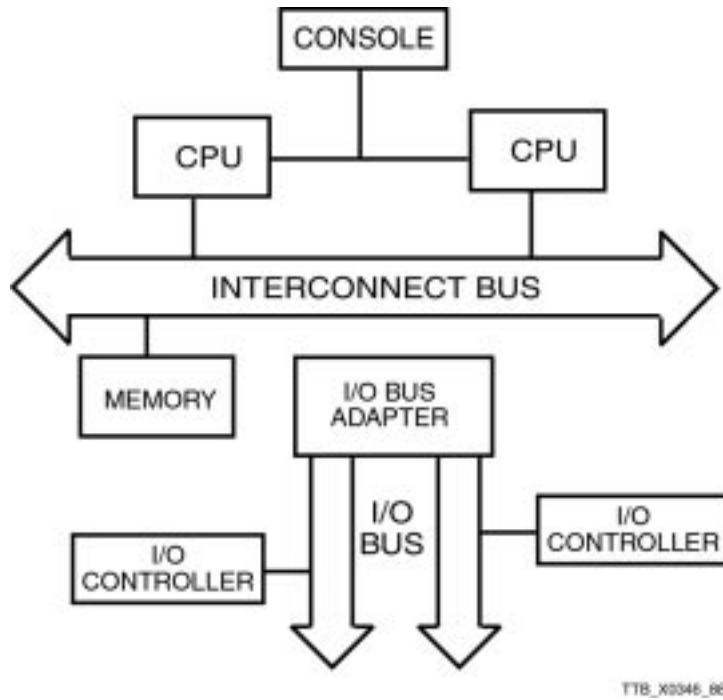


Figure Notes: Tightly Coupled System Configuration

In this configuration:

- Two CPUs share one console.
- An interconnect bus allows CPUs to share the same memory
- The pair of CPUs appears as one CPU when accessed through the common I/O bus by users and system managers.

OpenVMS Cluster Systems

Overview

An OpenVMS Cluster system is a loosely coupled configuration of computer systems and storage subsystems. Multiple systems running the OpenVMS operating system can be linked together, regardless of their size or capacity. Peripherals, such as mass storage devices and printers, are shared. An OpenVMS Cluster system appears as a single system to the user.



Note

OpenVMS Cluster systems are discussed in more detail in a later section.

Characteristics of an OpenVMS Cluster System

An OpenVMS Cluster system:

- Consists of cooperating nodes
- Uses OpenVMS software
- Is connected by a high-speed communication medium to share data and resources
- Can be configured with no single point of failure



Note

You may want to emphasize a few basic points

- Integrity of data
 - High availability
 - Flexibility of configuration
 - Single focus for system management
-

Loosely Coupled Network Configurations

Overview

A network is a configuration in which two or more devices are connected to enable information transfer. In the context of computing systems, a network is a collection of interconnected individual computer systems.

Characteristics of a Network

A network:

- Consists of two or more communicating nodes.
- Can consist of OpenVMS systems that can be connected to:
 - Other Digital systems
 - Other manufacturers' systems
 - A Local Area Network (LAN) – limited geographical area
 - A Wide Area Network (WAN) – larger area
 - A Metropolitan Area Network (MAN) – Large area of coverage within a metropolitan area like a city and surrounding communities

A network makes distributed resources available to a similarly distributed set of users.

Comparison of Multiple Processor Configurations

The following table summarizes characteristics of multiple processor configurations.

	Tightly Coupled <-----> Loosely Coupled		
System Characteristic	Multiprocessor	OpenVMS Cluster System	Network
CPU booting	Together	Separate	Separate
CPU failure	Together	Separate	Separate
CPU cabinet location	Single or adjacent	Same local area	Can be widely separated
Security domain	Single	Single	Multiple
Management domain	Single	Single	Multiple
Operating system	Shared – OpenVMS	Separate – all OpenVMS	Separate – some may not be OpenVMS
File system	Integrated	Integrated	Separate
Growth potential	Limited	Large	Very great

OpenVMS Cluster Systems

Overview

An OpenVMS Cluster system is a loosely coupled configuration of computer systems and storage subsystems. Multiple systems can be linked together, regardless of their size or capacity. Peripherals, such as mass storage devices and printers, are shared. An OpenVMS Cluster system appears as a single system to the user.

Topics

This section discusses the following topics:

- Members of a cluster
- Interconnects
- Sample OpenVMS Cluster configurations
- Connection manager
- Device names in an OpenVMS Cluster system

Members of a Cluster

Members of a cluster:

- Share resources
 - Processing resources
 - Queues
 - Disk storage
 - Tape storage
- Can use a single security and management domain
- Boot and fail independently
- Communicate with all other OpenVMS Cluster processors to coordinate cluster activities
- Potentially perform I/O to any disk storage or tape storage subsystem in the cluster
- Execute a private copy of the OpenVMS operating system in memory

Special Members of a Cluster

Certain members in a cluster can be designated to carry out special functions or responsibilities, which include:

- Boot servers
- Satellites
- Disk servers
- Tape servers
- Ethernet members



Note

HSJ-based disks and tapes, as well as DSSI based integrated storage elements (ISEs), can be shared by all cluster members. Shared devices and queues may be local to particular systems. That is, they may exist on or be connected to a system and made available to the cluster. Tape "sharing" only means that various systems have access to the tape over time, one at a time until it releases the device for reallocation.

Boot Servers

Boot servers:

- Downline load satellite boot files
- Serve the system disk to satellites
- Contain:
 - Cluster common files for startup
 - Directory roots that boot satellites

Satellites

Satellite nodes:

- Are MicroVAX or workstation members
- Do not have local system disks
- Boot remotely over the Ethernet from the boot server node

Disk Servers

Disk servers:

- Allow other OpenVMS Cluster nodes access to disks to which those nodes do not have a direct connection
- Can serve:
 - Locally attached disks
 - HSJ disks
 - DSSI disks

Tape Servers

Tape servers:

- Allow other OpenVMS Cluster nodes access to tapes to which those nodes do not have a direct connection
- Can serve:
 - Locally attached tapes
 - HSJ tapes
 - DSSI tapes

Ethernet Members

Ethernet members of a cluster:

- Use local system disk
- May serve local disks to other members, thus acting as a disk server

Interconnects

The type of OpenVMS Cluster configuration is determined by the interconnect device used for the System Communication Services (SCS) software.

CI Bus

Characteristics of a Computer Interconnect (CI) bus include:

- Is a high-speed (70 megabits per second), serial dual-path bus
- Connects VAX processors and Storage units, in conjunction with the Star Coupler and the VAX and Storage unit CI port controllers

Network Interconnect

Characteristics of Network Interconnect (Ethernet) include:

- Lower speed (10 megabits/second)
- Serial single-path bus
- Ability to transmit multiple protocols, DECnet and interprocessor SCS transmissions

FDDI Network Interconnect

Characteristics of an fiber distributed data interface (FDDI) Network Interconnect include:

- High-speed (100 megabits per second)
- Connections up to 40 Kilometers (25 miles)
- Ability to transmit multiple protocols, DECnet and interprocessor SCS transmissions

DSSI Bus

Characteristics of a DSSI bus include:

- Short (6 meters)
- Fast (32 megabits per second)
- Parallel bus

ATM Connections

LANs and Classical IPs can be connected over ATM networks.

MI Bus

A mixed interconnect (MI) bus is any combination of the interconnects.

Sample OpenVMS Cluster Configurations

OpenVMS Cluster Configurations

An OpenVMS Cluster configuration is characterized by the type of interconnect used for communication among its components.

This section gives examples of these OpenVMS Cluster configurations:

- CI OpenVMS Cluster system
- Ethernet OpenVMS Cluster system
- FDDI OpenVMS Cluster system
- DSSI OpenVMS Cluster system
- Mixed-interconnect OpenVMS Cluster system

CI OpenVMS Cluster System Configuration

A CI OpenVMS Cluster system is made up of OpenVMS members that communicate over the CI bus.

Figure: CI OpenVMS Cluster System Configuration

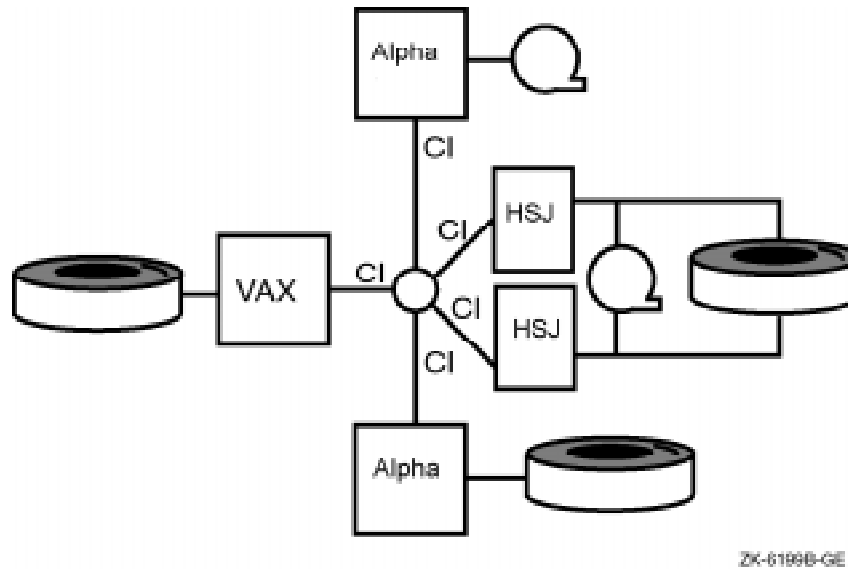


Figure Notes: CI OpenVMS Cluster System Configuration

This CI OpenVMS Cluster system configuration shows three computers.

- Two have local disks.
- One has a local tape.
- There are two disk controllers with a dual-ported disk and tape drive connected to them.

Any of the three computers in this OpenVMS Cluster system can mount the disk and tape that are connected to the storage controllers.

- Storage disks are more available to users than local disks because storage disks are not dependent upon the availability of any processor.
- The two local disks (connected directly to the VAX and Alpha computers) can be made available to any of the VAX systems, but if a computer with a local disk fails, that disk is no longer available to the rest of the OpenVMS Cluster system.
- The local tape (connected directly to an Alpha system) can be made available to any of the computers, but if a computer with a local tape fails, that tape is no longer available to the rest of the OpenVMS Cluster system.
- The disks are dual-pathed to the storage disk controllers, further increasing the disks' availability in the OpenVMS Cluster system.

If one storage disk fails, all traffic to the connected disks automatically fails over to the second controller.

Ethernet OpenVMS Cluster System Configuration

An Ethernet interconnect OpenVMS Cluster system is made up of a set of OpenVMS members that establish cluster connections and communicate using an Ethernet physical connection:

- MicroVAX and workstation satellites that boot by means of the Ethernet
- Boot servers that downline load the operating system and allow satellites access to their system disk

Ethernet interconnect OpenVMS Cluster components:

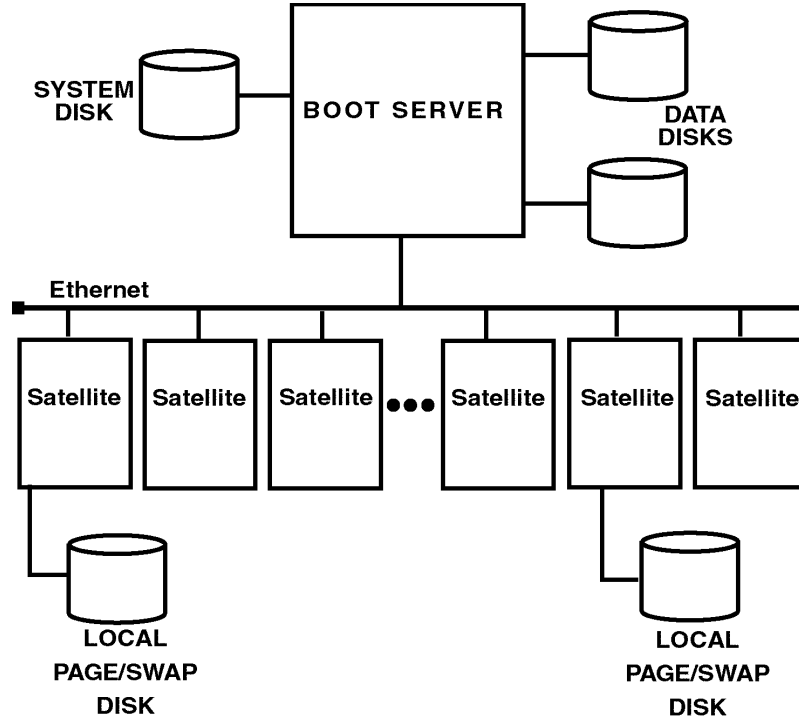
- Boot server
 - Management center and resource provider
 - System disk contains the system directory roots and cluster common files
- DECnet system
- DECnet Maintenance Operations Protocol (MOP) responds to satellite requests for downloading of OpenVMS system
- Satellites are generally consumers (not providers) of cluster services



Note

Ethernet OpenVMS Cluster system advantages include:

- Relatively low cost for OpenVMS Cluster benefits.
 - Hardware is also used for networking.
 - Ability to have wide geographic distribution.
 - Several separate clusters can coexist over the same Ethernet.
 - Disk space optimization versus each satellite storing identical system files.
 - Centralized system management, instead of individually managed systems.
 - Large VAX boot servers can offer high-performance batch queues.
 - Host-based shadowing is available.
 - Ethernet OpenVMS Cluster system disadvantages:
 - I/O throughput is limited by Ethernet adapters and Ethernet traffic
 - Network Interconnect SCS (NI-SCS) is more CPU intensive
 - Possibility of system disk bottleneck
 - The Ethernet cable is a single point of failure
-

Figure: Ethernet OpenVMS Cluster System Configuration

TTB_X0348_88

Figure Notes: Ethernet OpenVMS Cluster System Configuration

This Ethernet OpenVMS Cluster system configuration shows a number of VAX systems connected by way of Ethernet local area network connections.

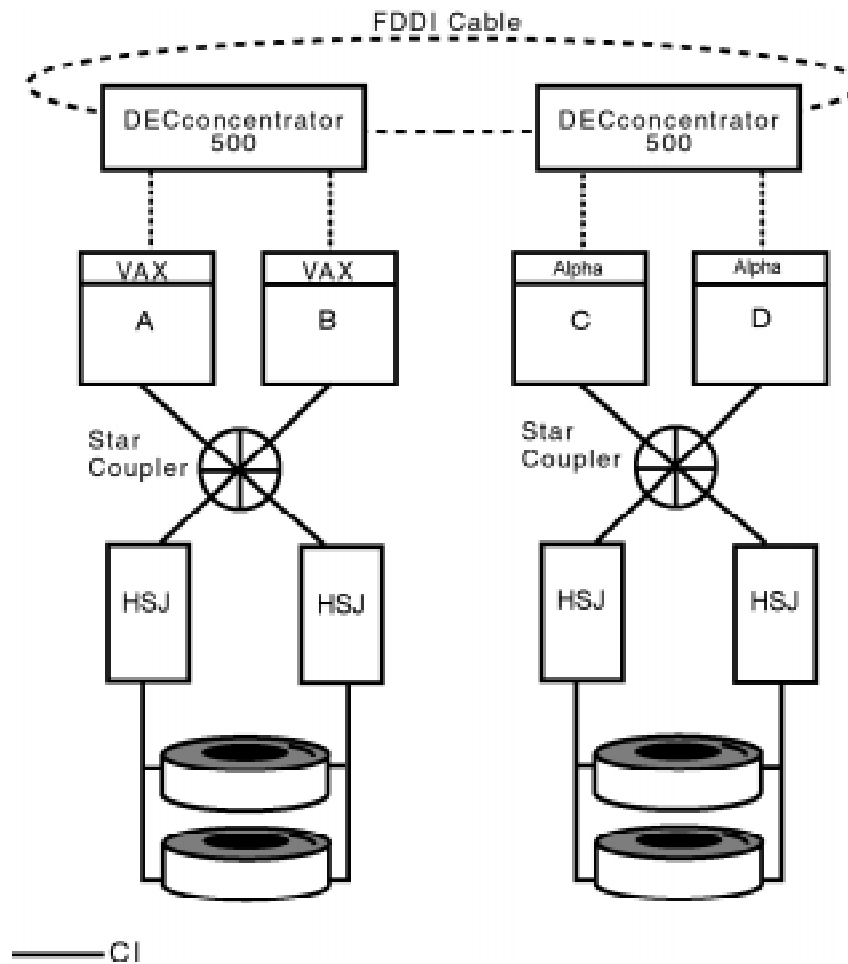
- The boot server serves files to the other computers (called **satellites**) in the cluster. A satellite downloads the OpenVMS operating system from the boot server, and consequently does not use a locally attached system disk.
- The boot server also functions as a disk server, making disks attached to the node available to the satellites. A satellite may have local disks. It can make these available to other cluster members, thus becoming a disk server on its own. The same is true for any local tapes attached to the boot servers or satellites.

FDDI OpenVMS Cluster System Configuration

The Fiber Distributed Data Interface (FDDI) interconnect utilizes fiber-optic cables for very high bandwidth over long distances.

A **lobe** is a collection of VAX and Alpha nodes connected by a CI and star coupler or connected by a DSSI. Such a collection could exist on its own as an OpenVMS Cluster system. When two or more lobes are connected by the FDDI, they become one large, multilobe OpenVMS Cluster system.

Figure: FDDI OpenVMS Cluster System Configuration



ZK-62009-GE

Figure Notes: FDDI OpenVMS Cluster System Configuration

This configuration is a multiple lobe FDDI OpenVMS Cluster system.

- The ellipse connecting the two DECconcentrator 500 units represents the FDDI cables.
- Nodes A and B make up one lobe of the OpenVMS Cluster.
- Nodes C and D make up the second lobe of the OpenVMS Cluster.
- The FDDI cable connects both lobes into a single OpenVMS Cluster system.



Note

The two lobes of the FDDI OpenVMS Cluster could be located at some distance from each other. This allows distant nodes to be part of the same OpenVMS Cluster giving them the resource sharing and management advantages of OpenVMS Clusters.

DSSI OpenVMS Cluster System Configuration

The DSSI bus allows certain processors and ISEs to communicate. More than one processor can reside on the interconnect, thereby providing more than one path to the ISE. The dual-host configuration allows members to communicate using the DSSI interconnect for disk access and cluster (SCS) communications.

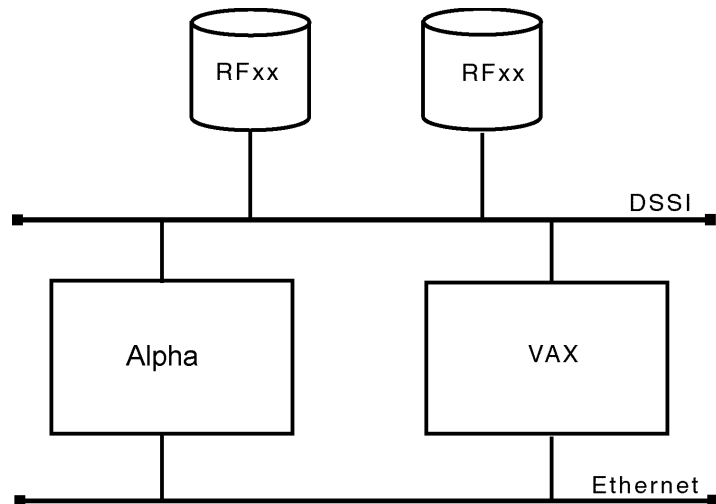


Note

Features of a DSSI OpenVMS Cluster system configuration:

- DSSI interconnect
 - ISE which acts as both controller and storage
 - KFQSA and embedded DSSI adapter (EDA) for DSSI interconnect to MicroVAX connections
-

Figure: DSSI OpenVMS Cluster System Configuration



ZKO-055-000149-07A-RGS

**Note**

Dual-host configuration advantages:

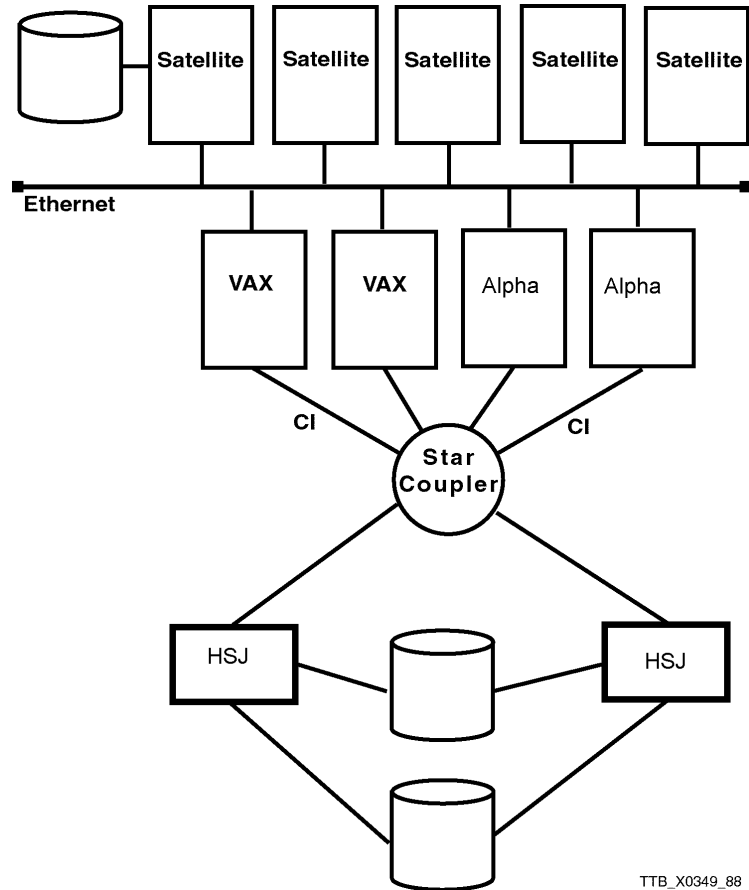
- Combines storage and controller functionality
- Eliminates contention for controller resources resulting in better I/O performance
- Centralized system management

Lower cost

- Dual-host configuration disadvantages:
 - Limited distance between DSSI nodes
 - Limited number of host systems supported
-

Mixed-Interconnect OpenVMS Cluster System Configuration

An OpenVMS Cluster system configuration can be very similar to a network configuration. An OpenVMS Cluster system can contain the same processors, communication devices, transmission media, terminal servers, and DECnet software as a network. However, the major difference between an OpenVMS Cluster system and a network is the OpenVMS Cluster software that synchronizes access to shared resources, such as data files, disks, tapes, and printers in an OpenVMS Cluster system.

Figure: Mixed-Interconnect Configuration**Figure Notes: Mixed-Interconnect Configuration**

This system configuration shows a combination of the other OpenVMS Cluster system configurations. It is possible with this type of configuration to have all disks on each VAX and Alpha system available to all other VAX systems in the OpenVMS Cluster system.

Connection Manager

One of the OpenVMS Cluster software components is the connection manager. The connection manager software runs on each active member in an OpenVMS Cluster to coordinate the state of the OpenVMS Cluster as members join or leave.

Functions of the Connection Manager

The connection manager:

- Determines what nodes are active in the OpenVMS Cluster
- Allows each active node to be aware of OpenVMS Cluster membership
- Conducts OpenVMS Cluster state changes
- Provides the necessary membership support to the lock manager
- Constructs and maintains the data structures needed to ensure a uniform view of the OpenVMS Cluster from each node
- Delivers messages to other remote nodes
- Removes nodes
- Uses a quorum scheme to prevent OpenVMS Cluster partitioning
 - Partitioning occurs when two or more OpenVMS Clusters have access to common resources and are unaware of each other
 - Partitioning can cause disk file corruption, since there must be coordination among all OpenVMS Cluster members sharing a file system
- Provides a highly available message service in which other software components can synchronize access to shared resources

Quorum

Cluster Quorum uses a dynamic value calculated by the connection manager to prevent partitioning. It allows processing to occur only if a majority of the expected voting member nodes are functioning.

Quorum Disk

The quorum disk acts as a virtual node, and is given votes to add to the cluster votes total. It increases the availability of small configurations (notable two-member clusters).

For a quorum disk to be used:

- One or more nodes must have a direct connection to the disk. These nodes are called quorum disk watchers.
- The system parameter DISK_QUORUM on each quorum disk watcher is set to the disk's device name. It is left blank for the other nodes in the OpenVMS Cluster.
- The remaining nodes recognize the specified name by communicating with the quorum disk watcher.
- A satellite should never be a quorum disk watcher.

Device Names in an OpenVMS Cluster System

Naming Devices

Because nodes in an OpenVMS Cluster system can share devices, those devices must be named without redundancy or ambiguity.

Device Name Formats

Two formats are used in naming devices in an OpenVMS Cluster environment:

- **node\$device**
 - Used for devices that are directly connected to only one node.
 - Zero is the default value for the allocation class SYSGEN parameter, forcing this format (ALLOCLASS for disks, TAPE_ALLOCLASS for tapes).
 - Used to refer to devices on other nodes when specifying:
 - A terminal on another node to OPCOM
 - A printer on another node to the job controller
- **\$allocation-class\$device**
 - Allocation class is a parameter set on a node that serves disks or tapes. This is an HSJ, or ISE controller or a node running the MSCP server for disks or the TMSCP server for tapes.
 - Zero is the default.
 - Setting the ALLOCLASS parameter to nonzero (1-255) enables this format for disks.
 - Setting the TAPE_ALLOCLASS parameter to nonzero (1-255) enables this format for tapes.
 - Used for dual-pathed (including dual-ported and dual-hosted) devices, to provide a single name for the device.
 - For any disk or tape device, all nodes that serve it to the cluster must have the same allocation class.

Duplicate Device Names

No two devices in an OpenVMS Cluster can have the same name.

For example, if two nodes have a disk allocation class 1, you cannot connect a drive named DUA0 to each node because there would then be two devices named \$1\$DUA0. To avoid this problem, change the unit number of one of the drives.

Naming Devices on a Local Node

When a device is on a local node:

- You can use its traditional name (for example, TXA2).
- You can prefix its name with the node name (for example, BARNUM\$TXA2).
- Each disk must have a unique volume label.



Note

The following terms are used in describing configurations.

This term...	Has this meaning:
Dual-ported	The disk has two cables coming out of the back. This means that the software and hardware can achieve better performance because of the interleaved data. This also allows two nodes to have direct access to the disk.
Dual-pathed	The disk has cables going to two different computers, so if one path fails, the data is still accessible by the other path. (You must log in from the remaining node.)
Dual-hosted (or multiple-hosted)	In a mixed-interconnect OpenVMS Cluster, the disk is served to the satellite nodes by more than one CI node.

Figure Notes: Sample OpenVMS Cluster System

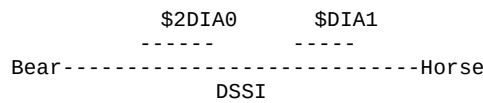
Note that there are three disks in this diagram that have the same unit number. (Example: DUA0: on LION.) If these disks are available clusterwide, the only difference in their names will be the node that serves them. (Example: BARNUM\$DUA0.)



Note

Note that all disks are dual-ported in this example. You may want to discuss how dual-porting works.

Consider redrawing the figure with the disks between BEAR and HORSE as DSSI devices, named \$2\$DIA0 and \$2\$DIA1:



Note that this figure shows only the nodes in the CIRCUS cluster. There might be other nodes on the same Ethernet, but they would not be able to mount these devices directly because they are not in the OpenVMS Cluster.

Network Configurations

Overview

A network is an example of a loosely coupled system. A network can consist of two or more communicating processors. An OpenVMS system can be connected to other Digital systems, or to other manufacturers' systems, to establish a network.

Topics

In this section the following topics are discussed:

- Network concepts
- Network components
- Network management

Network Concepts

Definition of Network

A **network** is an organized system of two or more computers that are linked together either locally or remotely to communicate and share information.

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

Need for Networks

Networks evolved to meet the growing needs of the enterprises that use computer systems.

- A computer system allows users to process, store, and manipulate data.
- A typical timesharing computer system supports numerous users, but user demands often exceed the capability of a single system.
- Companies, corporations, and organizations tend to have numerous computer systems, including personal computers and workstations, for users at their worksites or for various departments within a business.
- Two or more computer systems can be connected in a network to allow users (and departments) to:
 - Share access to common resources, such as storage, input/output, and communication devices
 - Share access to information and applications
 - Communicate between systems

Responding to Growing Needs

Typically, two trends drain the resources of a computer system: (1) the number of users increases, (2) users find new (more) uses of the computer system. To respond to these growing needs, one can add more computer power by performing one of the following actions:

- Increasing the capacity of the computer (such as a central computing facility).
- Adding a new computer of the same size and manufacturer.
- Placing more computing at the hands of the user (such as distributed computing, often with smaller, less expensive computers).

This could be a less expensive and more efficient solution. Users have direct, relatively quick access to computing power.

For example, all computers could access the same files. Some computers could function as file **servers**, offloading computing required for file **services** from client (user) computers. Servers share their services among the users of the network. Users of PCs and workstations can access data from any location on the network; users do not have to find a particular floppy disk in a particular desk drawer. Freed of file maintenance responsibilities, the client computers then have more computing power and better response time available for their users. In addition, client computers do not require disks of their own to store files.

Any computing resource can become a resource to the entire organization simply by attaching it to the network.

Nodes

The computer systems or intelligent devices that communicate with other devices in a network are called **nodes**.



Note

A node is a point in the network where service is provided and used, or communication channels are interconnected. A node is equipped with a central processor, network software, and communications hardware.

What the User Can Do Over the Network

DECnet software is designed to be transparent to the network user. You can perform general user operations over the network as easily as at the local node.

These operations include:

- Sending electronic mail to any user on the network
- Logging in to another network node on which you have an account
- Accessing common or public directories and databases located on any node in the network
- Manipulating files on remote nodes using several Digital Command Language (DCL) commands including: TYPE, PURGE, DELETE, DIRECTORY, COPY, RENAME and SEARCH
- Running database software, such as Rdb/VMS or DBMS, over the network
- Submitting jobs to remote print and batch queues

Using the OpenVMS Station Management tool for user account management and print queue management

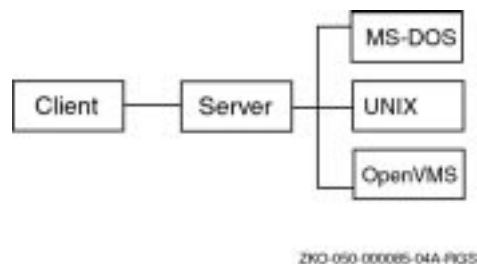
Client/Server Computing

Client/Server computing is a style of computing in which portions of applications are distributed across a network on server systems, which clients access to obtain needed computer-based resources. Clients make requests for services, and the servers respond with services. The client needs only to make the request.

Unlike the timesharing model, where the goal was to get as many clients attached to the host system as possible, this client/server model brings as many servers to the client as it needs. Based on Network Application Support (NAS), client/server computing will:

- Provide the functionality of complex, multivendor networks
- Maintain the simplicity of a single system

Figure: Client/Server Computing



Note

Client/Server computing focuses on the function of computers in a network rather than on the size of computers. It allows each portion of an application to execute in its own optimal environment. For example, display services are best provided by PCs, workstations, or windowing terminals. On the other hand, compute, database, and communication services are best provided by more powerful systems. Servers are combinations of hardware and software computer resources, performing specific tasks, that provide the needed capability through the network.

Any system may be a client or a server, and in some cases both may exist on the same platform. Typical client systems can include PCs, workstations, banking terminals, electronic cash registers, reservation terminals, robots, factory cell controllers, and laboratory instrumentation.

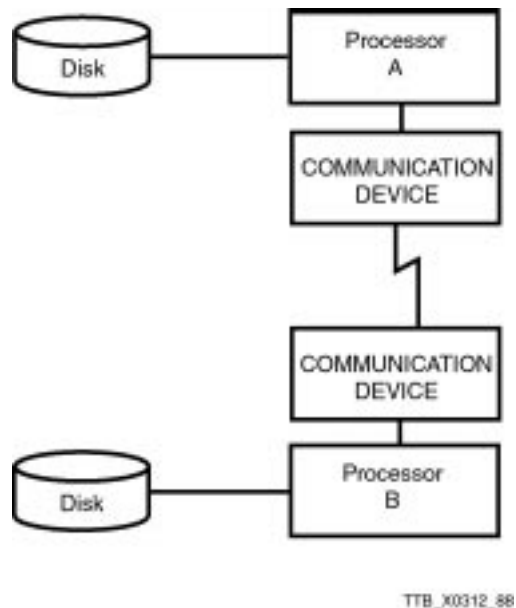
Servers are configured to support the needs of an organization. Existing general-purpose systems may be used as servers. Other examples of servers are systems optimized to perform the following tasks: database, transaction processing, file, communication, mail, printing, system management, and network management.

Network Components

The communication devices and data transmission media provide the physical connections between systems in a network. Certain devices and media are designed for use in local area networks, others for wide area networks.

Terminal servers provide connections between users' terminals and the systems in a network. A network configuration does not require terminal servers; they are optional. However, terminal servers provide increased availability of network resources to users.

DECnet software enables communication between the networked systems. This communication allows a user who is logged in to one system to communicate with users on other systems, and access data files on other systems.

Figure: DECnet Network**Figure Notes: DECnet Network**

- This network consists of two processors, or nodes. Each node has a disk drive.
- Each processor in the network has an attached communication device. The communication devices are connected by a data transmission medium.
- Because the two nodes are connected in a network, a user logged in to one of the nodes can:
 - Communicate with a user who is logged in to the other node
 - Access disk files stored on the other node
 - Write programs that communicate with programs running on the other node
- Access to disk files stored on a given node depends upon the availability of that node and the access rights to the disk files.

For example, if Processor A is shut down, any disk files stored on Processor A's disk become inaccessible to users logged in to Processor B.

Network Management

Network nodes cooperate with each other to share resources. This cooperation includes such features as:

- Communication paths
- A system of addressing and routing
- Consistent configuration of parameters affecting communication and network performance

Network management is the process that ensures this cooperation. This process is a combination of functions performed by network managers using features provided by software.

Network Management Functions

Network management functions are ensuring system integrity and include:

- Testing
- Controlling
- Monitoring
- Ensuring Security



Note

Network management combines people skills with software functions. The people provide the standards, guidelines, procedures, and organizational ability. The software provides the utilities and processes for implementation of those standards and procedures. These standards and procedures should create a viable resource-sharing mechanism.

Ask the students to name some of the functions they perform as network managers. Which categories (administrative, operational, technical) do they fall into? Use this as a lead-in to the subsequent discussion of network management functions.

Three Levels of Network Management

Depending on the size of the network, the network manager's duties may consist of three levels:

- Multinode network management
- Area network management
- Multiarea network management

As the network grows larger, the manager's responsibility grows broader and less specific. In fact, every network manager has the responsibilities allocated to all three levels. The division of duties comes when the network becomes too large for one person to handle.

Multinode Network

A multinode network consists of more than one node.

- Could be a small network or a subset of an area within a larger network.
- Can usually be managed by one person.

As remote nodes are added, management becomes more difficult because someone must be on site to manage individual nodes.

Area Network

An area network consists of many nodes within a single area address.

- This type of network has essentially the same configuration as a multinode network but may be too large for one person to manage.
- It can also cover the case where the management responsibilities of a very large network are divided by area.

Multiarea Network

A multiarea network is a very large network divided into many areas. The network manager handles all areas within a major network.

OpenVMS Software Concepts

Overview

The OpenVMS operating system performs three major functions:

- Provides communications between users and devices
- Creates a protected working environment for each user
- Schedules the use of system resources among all users in a fair manner

Many operating system activities happen without the user knowing anything about them. Others happen only when the user issues a command.

You will not be aware of many of the activities of the operating system in a properly configured system. They occur because the system initiates them. Other activities occur because you request them by entering commands at your keyboard. (You can see these activities or the results of them.) You express these commands in an English-like language known as the Digital Command Language (DCL). A special program called the command language interpreter (CLI) translates the commands. The interpreter calls the OpenVMS routines that execute your request.

This section discusses:

- Programs, images, and utilities
- Processes
- System files, directories, and logical names
- System file and directory protection

Programs, Images, and Utilities

Programs

A **program** is:

- Also called a **source program**
- A logical sequence of instructions
- A text file written in human-readable form (computer language)



Note

Strictly speaking, a program is a logical sequence of instructions written in a computer language that tells the computer how to perform a task, or set of tasks. This source program must be translated from human-readable form (text) into machine code, and then be linked to become a machine-executable image, as illustrated in the figure.

Quite often people use the terms **program** and **image** interchangeably. When the term **program** is used to refer to a functional piece of software, the implication is almost always an executable **image** comprising the machine-readable form of the program.

This use of the term **program** is so pervasive among computer users that this course henceforth uses **program** in the same ambiguous manner. For system management purposes, this ambiguity does not present a problem, because system managers usually deal with executable images and not source programs.

Object Files

An **object file** (filename.OBJ) is:

- A file containing one or more object modules
- Created as a result of translating or **compiling** the source program

Images

An **image** (filename.EXE) is:

- Also called an **executable image**
- A sequence of machine code instructions with addressing information
- Created as a result of **linking** object files together

Utilities

A utility:

- Is a program supplied with the OpenVMS operating system
- Sometimes has a command language all its own
- Typically uses DCL to call (invoke) a utility



Note

Programs typically execute the commands you specify at your terminal. The OpenVMS operating system includes a large number of programs that help you perform your data processing tasks quickly and effectively. You use many of these programs when you enter OpenVMS commands at your terminal. The DCL CLI translates the commands and calls the programs (invokes the images) that execute the commands.

Programs supplied with the OpenVMS operating system are often called utilities. Some utilities, such as text editors, have command languages of their own. You can invoke these utilities with the DCL command that calls the utility, and then use the utility's own command language.

Figure: Translating a Program into an Image

The source program is compiled or translated to create an object program, and the object is linked to create an image.



TTB_X0356_88

Processes

The working environment on an OpenVMS system is defined in terms of the **process**.

- All work on a system is performed within a process.
- The system uses information associated with processes to determine:
 - Order in which jobs are executed
 - Availability of system resources for a particular job
 - Allocation of those resources
- A process contains identification and status information that the system needs to execute programs.



Note

Installing images with the /SHARE qualifier allows pages of the program to be shared by processes and enhances image activation time by allowing a program to stay in memory after it has been run.

Types of Process

There are four types of process:

- Batch process
- Detached process
- Network process
- Subprocess

**Note**

This is very elementary at this point. Remind students that a process shows up as a line in the SHOW SYSTEM output.

Detached processes and subprocesses can be either interactive or noninteractive. "Interactive" is a property, rather than a process type.

Types of OpenVMS processes include the following:

Type	Created By	Creating Code	Special Properties
Batch	Job controller	SUBMIT, \$SNDJBC, SYS\$CREPRC	Deleted upon logout, or at end of command stream; no password check.
Detached	Another process	RUN, SYS\$CREPRC	Survives deletion of its creator; may or may not be interactive.
Network	Network ACP (result of DCL command with node name specified)	SYS\$CREPRC	Deleted when no more logical links to service.
Subprocess	Another process (the owner)	RUN, SPAWN, LIB\$SPAWN, SYS\$CREPRC	Cannot survive deletion of owner. Some quotas are pooled with owner. May or may not be interactive.

Detached Process

A detached process:

- Is created when a user logs in
- Terminates when the user logs out
- Can be interactive
- Does not affect the resource quotas of another process

Subprocess

A subprocess:

- Is created and owned by another process, called the parent process
- Uses some of the same resources as the parent process
- Allows the user to have several programs executing at once
- May or may not be interactive

Batch Process

A batch process:

- Is created by the system or user to execute a command procedure
- Creates a log of its activities
- Is not interactive

Network Process

A network process:

- Is created by a remote system when its node name is specified in a DCL command
- Creates a log file called NETSERVER.LOG in a directory on the remote node
- Is not interactive



Note

A network process is created by a remote system when its node name is specified in a DCL command.

- This is not the only way a network process is created, but that is all we will say in this course.
 - A network process is created when a network connection is made to a system, as when you run VTX or NOTES to access a remote node.
 - The network process does not necessarily terminate when the connection goes away. It may stay around for a while in case it is needed again.
-

Creating Detached, Batch, and Subprocesses

There are several reasons to create these types of processes:

- To allow a long program to execute without tying up your terminal
- To access the system outside your current process without interrupting what you are doing
- To maximize the use of system resources



Note

Do not discuss this in great detail. It is important that students understand how and when to create each of these; and when they will be created for them by the system.

PQL_DWSDEF will override process quotas.

Note that the command RUN/DETACHED requires DETACH privilege. If the process issuing the command does not have this privilege, a subprocess is created, but no message is sent to the screen that this has happened.

Characteristics of a Detached Process

A detached process:

- Is created with the RUN/DETACHED command
`$ RUN/DETACHED PROG.EXE`
- Has the same user identification code as the current process
- Uses the same initial resource quotas as the current process

Characteristics of a Batch Process

A batch process:

- Is created with the SUBMIT command
`$ SUBMIT MYSTUFF.COM`
- Processes created with the SUBMIT command read their instructions from a DCL command procedure.
- Is not dependent on your process, and:
 - It logs into the system under your name and password
 - It executes everything in your LOGIN.COM file just as if you had logged in

Characteristics of a Subprocess

A subprocess:

- Is created with the SPAWN command
\$ SPAWN
- Executes commands so that they do not affect any other process, with one exception: commands that affect terminal characteristics (SET TERMINAL). These affect all subprocesses that display to that terminal.

To exit from a subprocess:

- LOGOUT stops the subprocess and returns control to the process you were in at the time you spawned.
- ATTACH exits the subprocess and puts it into hibernation. You can then reattach to that process later.

You must supply the ATTACH command with either a process name or Process Identification Number (PID)



Note

Note that output from a subprocess would affect the other process.

Example: Spawning a Subprocess

The following example shows to spawn a subprocess:

```
❶ DIRECTORY/SIZE
Directory WORK7:[GERNELL]
AETNA.DIR;1          1
ANNOUNCE.TXT;3       1
ART.DIR;1            1
BATCH_QUEUE_CHECKER.;1
                      13

❷ [Interrupt]
❸ $ SPAWN
%DCL-S-SPAWNED, process GERNELL_1 spawned
%DCL-S-ATTACHED, terminal now attached to process GERNELL_1
❹ $ MAIL
MAIL> DIRECTORY                                NEWMAIL
# From          Date          Subject
1 SILLY::B_BEINKE 13-AUG-1999 Course Review
2 LSKLHK::GERNELL 15-AUG-1999 New manual
3 WAYOUT::SPEAKS  15-AUG-1999 Technical Review
4 LHKLSK::MENDEN  15-AUG-1999 Need some information
.
.
.
MAIL> EXIT
❺ $ LOGOUT
Process GERNELL_1 logged out at 15-AUG-1999 12:37:51.49
%DCL-S-RETURNED, control returned to process GERNELL
❻ $ CONTINUE
CMS_HANDOUT.SDML;7    37
COMMANDS.DIR;1        3
.
.
.
```

Example Notes: Spawning a Subprocess

From the previous example:

- ❶ This command produces a large amount of output.
- ❷ Interrupt the output by pressing Ctrl/Y.
- ❸ Issue the SPAWN command to create a subprocess. Note the system messages.
- ❹ Enter the MAIL command, which is executed within the subprocess.
- ❺ Log out of the subprocess. Control is returned to the parent process.
- ❻ Issue the CONTINUE command to resume the original command. Press Ctrl/W to return to the original display.

System Files, Directories, and Logical Names

To perform operations with files on an OpenVMS system, you must specify the location of those files. Devices in an OpenVMS system are named according to a standard format. System directories are referred to by system wide logical names.

OpenVMS Device Names

On some systems, device names follow the format **ddcu**:

dd	A two-letter device code
c	A one-letter code that specifies the hardware controller for the device. (Controllers provide the interface between the bus and the device, or between two buses.)
u	The device unit number

The device code specifies the device type.

The hardware controller number:

- Identifies the device controller
- Is represented by a letter from A to Z
- Is assigned by the system

The unit number:

- Indicates the position of the device on the controller
- Can be changed by:
 - Setting a button or switch on the device
 - Installing a unit plug on the device



Note

Local DSA devices use the controller letter "A," regardless of the physical controller on which the device resides. All local DSA disk devices are named **DUA n** , where n is a unique disk unit number.

If the device is attached to a single computer or storage disk subsystem, the device name includes the node name in the form **NODENAME\$DDCU:**, where **node** refers to the node name of the system on which the device resides.

Some Common Device Codes

Code	Device
CS	Console storage device
DU	Disk drive
LP	Line printer
MU	Tape drive
TT	Locally connected terminal

**Note**

A SHOW DEVICE command will probably display devices with names similar to \$1\$DUA1. The \$1\$ in front of the device name is determined by the allocation class of the device. Allocation class is a method that OpenVMS uses to make certain that devices have unique names known by all members of a cluster.

System Files

The OpenVMS operating system stores its image files, data files, and other files in special locations in auxiliary storage:

- **System volume**
Disk volume that holds system files
- **System device or system disk**
Disk drive on which the system volume is mounted
- **System directories**
Directories on the system volume that store the system files

System Directories

Each system directory has its own logical name, which specifies the system device and the specific system directory.

To use the directory name to refer to a file in a system directory, use the logical name `SYS$SYSROOT` as the device name, which is defined as a concealed root directory.



Note

By using the system logical name `SYS$SYSROOT`, an OpenVMS system can have a set of directories containing node-specific files, and a set of directories containing files shared with other nodes booting from the same system disk.

For example, to refer to the file `LINK.EXE` in the system directory `[SYSEXE]` in the system root, use the following file specification:

`SYS$SYSTEM:LINK.EXE`

To use the directory name to refer to a file in a system directory, refer to the system volume by the logical name `SYS$SYSROOT`. In the example above, you could use the following file specification:

`SYS$SYSROOT:[SYSEXE]LINK.EXE`

Mention that you will be talking about concealed root directories shortly, right after a quick overview of current system directories. The OpenVMS operating system uses primary and secondary file references to help structure its system disk.

- Note confusing terminology between file alias and node alias: for example, `SYSDUMP.DMP`
- To make a single dump file available to multiple nodes in a cluster, you can use primary and secondary file references
- The primary reference is made when you create the file
- Secondary references can be made using the `SET FILE/ENTER` command

If explaining primary and secondary file references in class, use an example like the previous one first, then show how they are used in OpenVMS software.

SYSSYSROOT

The logical name syntax for the system device and system directories implies that the logical name SYSSYSROOT is treated like a physical device such as DUA0. However, it is not that simple. If you use the SHOW LOGICAL command to translate the logical name SYSSYSROOT, it translates to a device and directory name, such as DUA0:[SYS0.]

In other words, you are not meant to enter the fully translated file specification for files in system directories.

Note that the logical name SYSSYSROOT is used for the system volume only when you refer to system directories. When referring to other directories on the system volume that do not involve a concealed root directory, use the logical name SYSSYSDEVICE for the system volume. This logical name translates to the device name of the system device, such as DUA0.



Note

The reason system directories are subdirectories of a first-level root directory is to allow multiple versions of the operating system to coexist on the same volume. Each system version has its own root directory of the form SYSSYSDEVICE:[SYSn].

The purpose of the logical name SYSSYSROOT and the logical names in the example is to conceal the existence of the first-level root directory [SYSn]. [SYSEXE] is actually a subdirectory of [SYSn]. The use of SYSSYSROOT frees users from needing to know the details of the directory structure (or even the system) that they are using.

Concealed Root Directories

Concealed root directories allow flexible handling of multiple versions of OpenVMS software on the same system.

System root directories are of the form **[SYSn]**, where n is a single hexadecimal digit from 0 through F. A typical nonclustered OpenVMS system will have only one system root directory: [SYS0].

Do not mix the use of root directory names and top-level directory names:

- **Correct:** SYS\$SYSROOT:[SYSEXE]LINK.EXE
- **Incorrect:** SYS\$SYSROOT:[SYS0.SYSEXE]LINK.EXE



Note

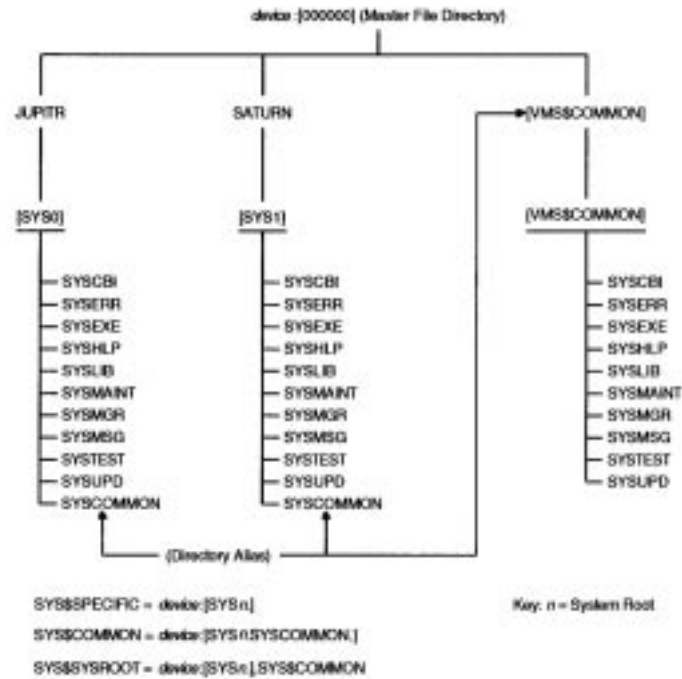
The example with the incorrect directory specification translates to a nonexistent directory, DUA0:[SYS0.SYS0.SYSEXE]. You cannot include a root directory name and a top-level directory name in the same directory specification. In this example, [SYSEXE] is a top-level directory on the pseudo-device DUA0:[SYS0.], i.e., SYS\$SYSROOT.

Under usual operating conditions, in a nonclustered environment, the only root directory used is [SYS0]. An alternate root directory can be specified at system startup, but this subject is beyond the scope of this course. Once the system starts, you can use the logical names discussed above without knowing the identity of the concealed root directory.

At the present time, the principal use of concealed root directories is to improve the reliability of upgrading an OpenVMS system.

Figure: System Directory Tree

The following figure shows a map of a system directory tree.



OpenVMS System Directories

The following table summarizes the contents of the OpenVMS system directories.

Directory Name Used in Combination with SYS\$SYSROOT	Logical Name and Principal Contents
[SYSEXE]	SYS\$SYSTEM Executable images, user authorization file (UAF), site-independent startup and shutdown command procedures, system parameter files, print form definitions, page file, swap file, crash dump file
[SYSLIB]	SYS\$LIBRARY and SYS\$SHARE Macro libraries, object libraries, shareable image libraries, shareable images
[SYSHLP]	SYS\$HELP Help libraries
[SYSMSG]	SYS\$MESSAGE Error message files
[SYSMGR]	SYS\$MANAGER Site-specific startup and shutdown command procedures, operator's log, accounting file
[SYSMAINT]	SYS\$MAINTENANCE Hardware diagnostics (not distributed with an OpenVMS system)
[SYSTEST]	SYS\$TEST UETP command procedures, images, data
[SYSERR]	SYS\$ERRORLOG Error log file
[SYSUPD]	SYS\$UPDATE Command procedures used for system installation, upgrade, and update
[SYSUPD.EXAMPLES]	SYS\$EXAMPLES Sample source programs such as device drivers and system services
[SYSCBI]	SYS\$INSTRUCTION Computer-based instruction software
[SYS\$LDR]	SYS\$LOADABLE_IMAGES Loadable executive images and device drivers
[SYS\$STARTUP]	SYS\$STARTUP Startup command procedures for site-specific utilities, software, and such



Note

You can find a list of all the files distributed in the system directories in the *OpenVMS Installation and Operation Guide* for your processor.

Briefly review the table and be sure all students are familiar with the purpose of each system directory. Mention that these directories (or the logical names that point to the system directories) will be mentioned repeatedly throughout this course.

Related System Logical Names

The following table shows some common system logical names that refer to system devices and directories.

Logical Name	Function
SYS\$SYSDEVICE	Points to the physical system device Example: \$ SHOW LOGICAL SYS\$SYSDEVICE "SYS\$SYSDEVICE" = "\$1\$DUA0:" (LNM\$SYSTEM_TABLE)
SYS\$SYSROOT	Rooted logical name for your node, points to both node-specific and shared files on the system disk, dev:[SYSn.] and SYS\$COMMON Example: \$ SHOW LOGICAL SYS\$SYSROOT "SYS\$SYSROOT" = "\$1\$DUA0:[SYS2.]" (LNM\$SYSTEM_TABLE) = "SYS\$COMMON:"
SYS\$SPECIFIC	Points to the node-specific portion of the system disk, dev:[SYSn.] Example: \$ SHOW LOGICAL SYS\$SPECIFIC "SYS\$SPECIFIC" = "\$1\$DUA0:[SYS2.]" (LNM\$SYSTEM_TABLE)
SYS\$COMMON	Points to the portion of the system disk shared by all nodes, dev:[SYSn.SYSCOMMON] Example: \$ SHOW LOGICAL SYS\$COMMON "SYS\$COMMON" = "\$1\$DUA0:[SYS2.SYSCOMMON.]" (LNM\$SYSTEM_TABLE)

System File and Directory Protection

System directories and files are distributed with sufficient protection to keep them safe from all users without a system UIC or elevated privileges. To maintain system security, this protection should not be changed.

It is important to understand the OpenVMS file protection mechanisms. If necessary, consult the command description for SET SECURITY in the *Security Guide*.

Be careful not to accidentally overprotect certain files. Users who need the files may not be able to get the needed access.



Note

Remember the following facts about protection:

- If a file contains an executable image, you can allow a user to run the image but not to make a copy of it by granting E access and denying R access.
- If you grant a user E access to a directory and deny R access, that user can look up files in the directory by specifying their exact file names and file types. However, the user cannot use a wildcard character to look up files in the directory. Note that the protection of the file itself also limits user access.
- Deny W access to all sensitive directory files. A user who has W access to a directory can create a file in the directory with the same name as a file already there, and with a higher version number.
- When a text editor creates a new version of a file, the editor assigns the file the same protection code as the previous version. Use the SET SECURITY command to change the protection code on the new version as needed.

When you create a new file, the system assigns your default protection code unless you specify a different code or an access control list (ACL) on the directory. Use the DIRECTORY/SECURITY command to determine the protection of a file; use SET SECURITY to set file protection.

Note that the protection of an edited file can become too limited as well as too unlimited. For example, system default protection denies all file access to the world category of users. But the system login file, SYS\$MANAGER:SYLOGIN.COM, must permit execute (E) access to World. If a process cannot read this file at login, the DCL commands in the file will not be executed.

Reminder: Protecting command files by allowing Execute access instead of simple Read access permits access only if the user knows the file specification. The definition of file specification is DIRECTORY:FILENAME.

Summary

Concepts

System Configurations

- A single processor configuration is a single independent VAX processor and its peripheral devices.
- In a tightly coupled configuration:
 - CPUs share operating system code
 - CPUs cannot operate independently from each other
 - Management and operation are handled as a single machine
 - The environment provides high performance
- An OpenVMS Cluster system is a loosely coupled configuration of computer systems and storage subsystems.
- In an OpenVMS Cluster system, peripherals such as mass storage devices and printers are shared.
- An OpenVMS Cluster system appears as a single system to the user.
- A network is a collection of interconnected individual computer systems.
- Multiple processor configurations range from multiple CPUs in the same cabinet (tightly coupled processors) to multiple CPUs widely separated (network processors).

OpenVMS Cluster Systems

- An OpenVMS Cluster system is a loosely coupled configuration of computer systems that share resources.
- The type of OpenVMS Cluster configuration is determined by the interconnect device used for communication.
- The connection manager coordinates the state of the OpenVMS Cluster as members join or leave.
- The quorum scheme allows processing to occur only if a majority of the expected OpenVMS Cluster members are functioning.
- In an OpenVMS Cluster system, devices are named in such a way as to prevent duplication or ambiguity of names.

Network Configurations

- A network allows users to share resources and information.

OpenVMS Software Concepts

- Understanding processes provides ways to manipulate and control the user environment.
- OpenVMS system files are kept in special system directories that can be accessed using default system logical names.

Developing Command Procedures

Module 2

About This Module

Introduction

The Digital Command Language (DCL) allows system users to communicate with the OpenVMS operating system. DCL provides an extensive set of commands to get information about the system and modify work environments. It also provides data manipulation and flow-control mechanisms similar to those found in programming languages, so that a command procedure can function as a sophisticated application program.

You can create DCL procedures to automate frequently performed tasks, such as maintaining user accounts, monitoring resources, and controlling security.

Objectives

To automate frequently performed tasks, the system manager should be able to write command procedures that:

- Access local and global symbol tables.
- Use symbols in command lines.
- Use terminal I/O to communicate with the user.
- Use symbols as variables or parameters.
- Control the flow of execution in a command procedure.
- Redirect input and output to a file.
- Obtain and manipulate information with commonly used lexical functions.
- Write command procedures that perform file I/O.
- Detect and handle errors and Ctrl/Y interrupts.

Accessing Local and Global Symbol Tables

Overview

When you log in to an OpenVMS system, you are at what is called the **DCL command level**. When you execute a command procedure, you create another command level. If that command procedure runs another command procedure, then yet another command level is created.

This structure is not unlike a hierarchy:

- DCL command level
 - First command procedure below DCL command level
 - ◆ Second command procedure below DCL command level
 - .
 - .
 - .

A process can have up to 32 of these command levels.

When you use symbols in command procedures, it is important to understand these levels and how local and global symbols work so you can pass information from one level to another using symbols.

Topics

In this section the following topics are discussed:

- Local and global symbols
- Terminating a command procedure

Local and Global Symbols

Two Kinds of Symbols

You can create two kinds of symbols:

- Local
- Global

Local Symbols

Local symbols:

- Are defined with either = or :=
- Are available to:
 - The command level that defined them
 - All lower command levels
- Are defined for the duration of the command level that defined them
- Can be explicitly deleted using the DELETE/SYMBOL command

Local Symbol Table

Each command level has a local symbol table. A local symbol table contains:

- The local symbols created in the command procedure
- The symbols P1 through P8 (stored as null character strings ("")) if values have not been supplied for them)

Global Symbols

Global symbols:

- Are defined with either == or :=
- Are available to all command levels of a process
- Allow you to pass information between command levels
- Defined for the duration of the process
- Can be explicitly deleted using the DELETE/SYMBOL/GLOBAL command

Global symbols can be used to pass information between command levels.

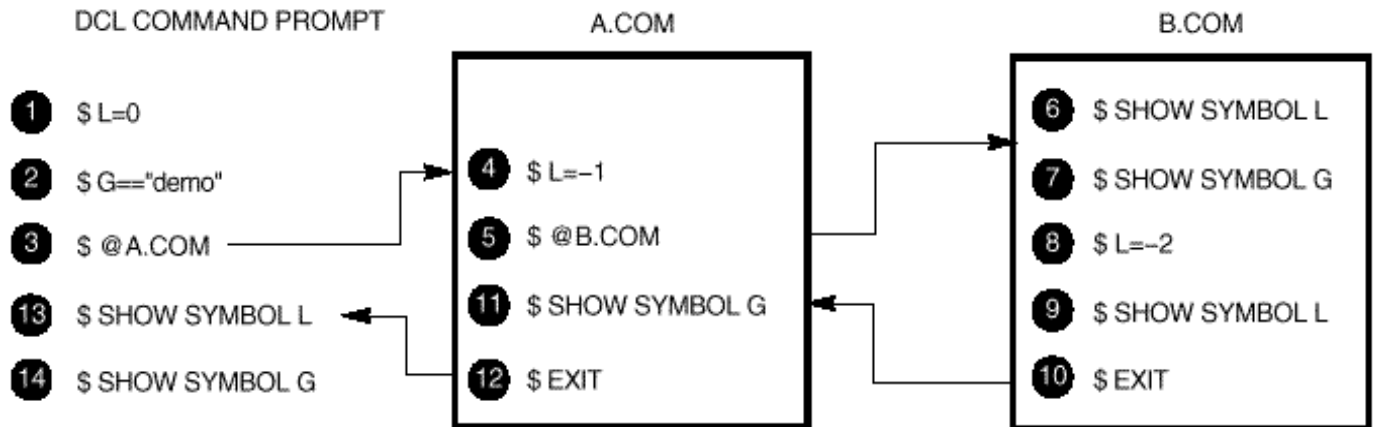
Global Symbol Table

There is one global symbol table for each process. It contains global symbols that you create and global symbols created by the OpenVMS operating system.

Example: Accessing Local and Global Symbol Tables

The following example displays local and global symbols defined and displayed at different command levels.

- At the DCL command level, the local symbol L and the global symbol G are defined.
- When the command procedure A.COM executes, it changes the value of L and then invokes the command procedure B.COM.
- B.COM displays the values of L and G and then changes the value of L.



Example Notes: Accessing Local and Global Symbol Tables

Notes on the Accessing Local and Global Symbol Tables example include:

- 1 Create a local numeric symbol L at DCL level
- 2 Create a global string symbol G at DCL level
- 3 Execute a command procedure, A.COM
- 4 Create a local numeric symbol L, equal to -1
- 5 Invoke another command procedure, B.COM
- 6 The symbol L equals -1
- 7 The symbol G equals "demo"
- 8 Create a local symbol L equal to -
- 9 The symbol L equals -2
- 10 Return control to the previous level
- 11 The symbol G equals "demo"
- 12 Return command to the previous level
- 13 The symbol L equals 0
- 14 The symbol G equals "demo"

Output from Example

Additional notes on accessing local and global symbol tables output include:

```
❶ $ L=0
❷ $ G=="demo"
❸ $ @A.COM
❹ L = -1 Hex = FFFFFFFF Octal = 3777777777
❺ G == "demo"
❻ L = -2 Hex = FFFFFFFE Octal = 3777777776
❼ G == "demo"
❽ $ SHOW SYMBOL L
    L = 0 Hex = 00000000 Octal = 0000000000
❾ $ SHOW SYMBOL G
    G == "demo"
$
```

Terminating a Command Procedure

The way a command procedure terminates determines the resulting command level.

This command...	Causes control to return to:
\$ EXIT or End-of-file	The previous command level
\$ STOP	DCL command level (command level 0)

Using Symbols in Command Lines

Overview

For a command procedure to use symbols, the OpenVMS operating system must translate symbols into their corresponding values. This is called **symbol substitution**. In certain contexts, DCL automatically evaluates symbols. Where automatic substitution does not take place, the command procedure must force the evaluation of the symbol.

Topics

In this section the following topics are discussed:

- Automatic symbol substitution
- Forcing symbol substitution

Automatic Symbol Substitution

In some situations, DCL automatically substitutes the value of a symbol.

Examples of Automatic Symbol Substitution

The following are common situations in which DCL automatically substitutes the value of a symbol.

Situation:	Sample commands:
Command synonym (first item after \$ prompt)	\$ XX = "DELETE/LOG" \$ XX FILE.TXT;
In an IF, WRITE, or INQUIRE command	\$ IF COUNT .GT. 10 THEN - WRITE SYS\$OUTPUT COUNT \$ INQUIRE NAME "Enter file name"
In the right hand side of an = or == assignment statement	\$ COUNT = COUNT + 1 \$ FILESPEC = NAME + ".TXT"

Forcing Symbol Substitution

To have symbol substitution occur in a situation where DCL would not normally perform symbol substitution, you must force symbol substitution.

How to Force Symbol Substitution

To force symbol substitution when DCL is not expecting to encounter a symbol:

- Enclose the symbol name in apostrophes (').
- Within a character string, precede the symbol with two apostrophes (") and end the symbol with a single apostrophe (').

When to Force Symbol Substitution

The following are common situations in which you must force symbol substitution.

Situation:	Sample command:
In a DCL command that does not perform automatic symbol substitution	\$ RUN 'PROGRAM'
In a character string	\$ WRITE SYS\$OUTPUT - "The file 'FILE' exists."
When concatenating two symbols in a DCL command that does not perform automatic symbol substitution	\$ PRINT 'NAME'.'TYPE'

Example: Performing Symbol Substitution

The following example shows how to use symbol substitution:

```

❶ $ TYPE REPORT.COM $!          REPORT.COM
  $! Print out a report for the day of your choice, located
  $! in the subdirectory that corresponds to that day.
  $
  $! Ask which daily report to print out
❷ $ INQUIRE DAY "Day to print a report"
❸ $ WRITE SYS$OUTPUT "Printing the ''DAY' report"
  $
  $! Print out the report for the correct day
❹ $ PRINT REPORT_DISK:[REPORTS.'DAY']'DAY'.RPT
  $ EXIT
❺ $ @REPORT
❻ Day to print a report: TUESDAY
❼ Printing the TUESDAY report
❽ Job TUESDAY (queue SYS$PRINT, entry 47) started on
WORK$TXA0
  $

```

Example Notes: Performing Symbol Substitution

Notes on the Performing Symbol Substitution example include:

- ❶ Display the contents of REPORT.COM.
- ❷ The local symbol DAY will be equated to the user's input.
- ❸ Substitute the value of DAY in a character string. Use two apostrophes to force substitution. The third apostrophe is optional because a space follows the symbol name.
- ❹ Symbol substitution is forced twice in this command. In each pair of apostrophes, the second apostrophe is required because non-space characters follow the symbol name.
- ❺ Invoke the command procedure.
- ❻ Command ❷ is executed.
- ❼ Command ❸ is executed.
- ❽ Command ❹ is executed.

Writing Command Procedures That Perform Terminal I/O

Overview

Terminal I/O allows a command procedure to interact with the user. Terminal input and output can be used to:

- Prompt the user for information
- Display messages and command output on the terminal screen
- Allow the use of an interactive utility, such as an editor
- Redirect terminal output to a file

Topics

In this section the following topics are discussed:

- Logical names used in terminal I/O
- Displaying information at the terminal
- Getting information from the user

Logical Names Used in Terminal I/O

System-Defined Logical Names

Several system-defined logical names are created by the OpenVMS operating system when you log in. Because they remain defined for the life of your process, they are referred to as **process-permanent** logical names. These logical names can be redefined in command procedures and interactive commands that affect I/O operations, but they cannot be deleted.



Note

You control terminal I/O by redefining the same logical names that DCL uses to obtain and display information for your process. These logical names are used interactively to communicate with your process when you are typing at your terminal.

In similar fashion, you can use logical names in a command procedure to enable communication between the command procedure and the system.

Commonly Used Process-Permanent Logical Names

The operating system creates several logical names for each process. Four of these logical names are commonly used in command procedures.

SYS\$INPUT	The default device or file from which DCL reads input.
SYS\$OUTPUT	The default device or file to which DCL writes output.
SYS\$ERROR	The default device or file to which DCL writes error messages generated by warnings, errors, and severe errors.
SYS\$COMMAND	The initial file from which DCL reads input. (DCL uses this logical name to remember your initial input stream.)



Note

Before explaining how to control terminal I/O, you might want to explain the system logical names in general in the table. Once students understand how the system uses these logical names to communicate with an interactive user, they can better understand how they can be used by a command procedure.

Characteristics of Process-Permanent Logical Names

Process-permanent logical names are created by DCL when you log in and remain defined for the life of your process.

- You cannot delete these logical names.
- You can redefine these logical names by specifying the same name in a DEFINE or ASSIGN command, but if the name is later deassigned, the process-permanent name is reestablished.

Using Help for Command Procedures

You can get help at anytime on any command or function by typing in the word `HELP` at the DCL prompt. You will see a series of menus that narrow your choices until you find the information you need.

Displaying Information at the Terminal

DCL Commands for Displaying Information at the Terminal

There are two DCL commands that can be used within command procedures to display information at the terminal. These commands are:

- WRITE
- TYPE

WRITE Command

Use the WRITE command in a command procedure to display information at the terminal.

Format:

```
$ WRITE SYS$OUTPUT expression
```

The Expression Parameter

The *expression* parameter specifies data to be written as a single record. It can be:

- A character string enclosed in quotation marks

```
$ WRITE SYS$OUTPUT "Hello"
```
- A symbol name

```
$ WRITE SYS$OUTPUT X
```

(The symbol's value is automatically substituted.)
- A lexical function

```
$ WRITE SYS$OUTPUT F$TIME( )
```
- A combination of items separated by commas

```
$ WRITE SYS$OUTPUT "The sum is ",X
```

(The command interpreter concatenates the items into one record.)

Specifying Data Items

You can specify data items using character string expressions, which may be:

- Symbol names
- Character strings in quotation marks
- Literal numeric values
- Lexical functions

Example: Using WRITE SYS\$OUTPUT

The following code displays a simple user menu with the WRITE command.

```
❶ $ WRITE SYS$OUTPUT ""
❷ $ WRITE SYS$OUTPUT "      SELECT AN ITEM:"
$ WRITE SYS$OUTPUT ""
$ WRITE SYS$OUTPUT "  1. ENTER DATA AT THE TERMINAL"
$ WRITE SYS$OUTPUT ""
$ WRITE SYS$OUTPUT "  2. ENTER NAME OF DATA FILE"
$ WRITE SYS$OUTPUT ""
```

Example Notes: Using WRITE SYS\$OUTPUT

Notes on the Using WRITE SYS\$OUTPUT example include:

- ❶ Cause a blank line to be displayed.
- ❷ Display a character string by enclosing it in quotation marks.

Output from Example

Output from the WRITE SYS\$OUTPUT example is as follows:

```
      SELECT AN ITEM:
1. ENTER DATA AT THE TERMINAL
2. ENTER NAME OF DATA FILE
```



Note

You could create a symbol:

```
$ WS = "WRITE SYS$OUTPUT"
```

Then the code would look like:

```
$ WS " "
$ WS " "
$ WS " "
$ WS "      SELECT AN ITEM:"
.
.
.
```

TYPE Command

The TYPE command can be used to display several lines on the terminal.

Format:

\$ TYPE *file-spec*

- If you specify a file specification (or a logical name that represents a file specification) for the *file-spec* parameter, the contents of that file will be displayed at the terminal.
- If you specify SYSS\$INPUT as the *file-spec* parameter, DCL will display the lines of the command procedure that follow the TYPE command until a \$ is encountered in column 1.
- You cannot use TYPE SYSS\$INPUT to display the contents of a symbol.

Example: Using TYPE SYSS\$INPUT

The following example show how to use the TYPE SYSS\$INPUT command:

```
$ TYPE SYSS$INPUT
```

```
      SELECT AN ITEM:
```

```
      1. ENTER DATA FROM THE TERMINAL
```

```
      2. ENTER NAME OF DATA FILE
```

```
$
```

Example Output

This code produces the following output:

```
      SELECT AN ITEM:
```

```
      1. ENTER DATA FROM THE TERMINAL
```

```
      2. ENTER NAME OF DATA FILE
```


Getting Information from the User

DCL Commands for Getting Information from the User

There are two DCL commands that can be used within command procedures to get information from the user. These commands are:

- INQUIRE
- READ

INQUIRE Command

Use the INQUIRE command to get information from the terminal and to define a local symbol with the information.

Format:

\$ INQUIRE symbol-name ["prompt"]

Example:

```
$ INQUIRE NAME "Enter file name"
```

- The optional *prompt* is displayed at the terminal followed by a colon (:). If you do not want to display a colon (:) after the "prompt" string, use the /NOPUNCTUATION qualifier with the command.

```
$ INQUIRE/NOPUNCTUATION NAME "What is the file name? "
```
- The *symbol-name* parameter is a local symbol that is equated to the user's response. The input is converted to uppercase. Multiple blanks and tabs are replaced with a single space. Use INQUIRE/GLOBAL to create a global symbol.



Note

- The INQUIRE command creates a **local** symbol.
 - Use INQUIRE/GLOBAL to create a global symbol.
 - Emphasize that INQUIRE converts input to uppercase.
 - This command is specifically for terminal I/O.
-

READ Command

Use the READ command to get information from the terminal or from a file.

Format:

\$ READ[/PROMPT=string] SYS\$COMMAND symbol-name

Example:

```
$ READ/PROMPT="Enter filename" SYS$COMMAND NAME
```

- Use the /PROMPT qualifier to specify a prompt with the READ command. You must supply any spacing or punctuation you wish the prompt to include.
- The *symbol-name* parameter is a local symbol that will be equated to the user's response.
- The user's response is **not** converted to uppercase, and multiple spaces are **not** removed.

READ/PROMPT Command

The default prompt for a READ command is "Data: ".

Example:

```
$ READ SYS$COMMAND NAME
Data: TEST.DATA
$
```

Example: Terminal I/O in a Command Procedure

This command procedure illustrates the use of the INQUIRE and READ commands to get input from the terminal, and the WRITE and TYPE commands to display output at the terminal.

```

❶ $ TYPE IO.COM
$!      I O . C O M
$!      Demonstration of simple terminal I/O.
❷ $ TYPE SYS$INPUT
❸ Using INQUIRE
❹ $ INQUIRE YOU1 "Please type your name"
❺ $ WRITE SYS$OUTPUT YOU1," is using this terminal."
$ WRITE SYS$OUTPUT "" ! Blank line.
$!
$!
$ WRITE SYS$OUTPUT "Using READ"
$ WRITE SYS$OUTPUT ""
❻ $ READ/PROMPT="Please type your name" SYS$COMMAND YOU2
$ WRITE SYS$OUTPUT YOU2," is using this terminal."
$ WRITE SYS$OUTPUT ""
$ EXIT
❼ $ @IO
❽ Using INQUIRE
❾ Please type your name: System User
❿ SYSTEM USER is using this terminal.
⓫ Using READ

⓬ Please type your nameSystem User
⓭ System User is using this terminal.
$

```



Note

Points to make about the example:

- The person's name is stored in a **local** symbol (YOU1 or YOU2).
- Even though the input was in lowercase, the INQUIRE command converts it to uppercase. (Override this by enclosing input in quotes.)
- Note the blank line before the final dollar sign appears. (Caused by `WRITE SYS$OUTPUT ""`)
- This example randomly combined `TYPE SYS$INPUT` with `INQUIRE`, and `WRITE SYS$OUTPUT` with `READ`. Don't allow students to think they must be matched that way. Any/all combinations are fine.
- `TYPE SYS$INPUT` is typically used to display **multiple** lines of text. The example only shows it displaying one line, to keep the code simple.

Example Notes: Terminal I/O in a Command Procedure

Notes on the Terminal I/O in a Command Procedure example include:

- ❶ Display the contents of the command procedure.
- ❷ Use the TYPE command to display output at the terminal. Output from this command line is shown in ❸ in the example.
- ❸ This is a data line, so it does not begin with a dollar sign. The next line, a command line, begins with a dollar sign and terminates the data for TYPE. Typically, TYPE SYS\$INPUT is used to display multiple lines of text. For simplicity, this example only displays one line.
- ❹ Use the INQUIRE command to get input from the terminal. The execution of this command occurs at ❺ in the example.
- ❻ Use the WRITE command to display output at the terminal. Separate the symbol and the text with a comma. The output, shown in (10), is a combination of a symbol and some text.
- ❼ Use the READ command to get input from the terminal. The execution of this command occurs at (12) in the example.
- ❼ Execute the command procedure.
- ❽ Output from the TYPE command ❷ in the command procedure.
- ❾ INQUIRE adds a colon and a space to the prompt. (Suppress the colon by using INQUIRE/NOPUNCTUATION.)
- ❿ INQUIRE converts input to uppercase and reduces multiple spaces to one space.
- ⓫ This blank line is the result of WRITE SYS\$OUTPUT "".
- ⓬ Punctuation and spacing are not supplied by READ.
- ⓭ READ preserves the case and spacing of the input.

Using Symbols as Variables and Parameters

Overview

Symbols are commonly used as variables in command procedures. They can also be passed as parameters to a command procedure when the command procedure is invoked.

Topics

In this section the following topics are discussed:

- Using symbols as variables
- Using symbols as parameters for command procedures

Using Symbols as Variables

Symbols as Variables

One of the most common uses for a symbol in a command procedure is as a **variable**. Think of a variable as a place holder for a value that the command procedure needs for performing a specific task. For example, the symbol COUNT could be used as a counter variable in a loop, and tested to determine whether the loop has completed.

Performing Operations with Symbols

These operators can be used to perform operations with symbols:

Operator	Meaning and Example
+	String concatenation FILENAME + ".EXE"
-	String reduction FILE_SPECIFICATION - ".LOG"
+	Arithmetic sum COUNT + 1
-	Arithmetic difference FIRST - SECOND
*	Arithmetic product 5 * DAILY_CHARGE
/	Arithmetic division (integer quotient) TOTAL / 10

Using Symbols as Parameters for Command Procedures

Symbols as Parameters

In addition to any local symbols that you might create, the OpenVMS operating system creates eight local symbols for you. These symbols are named P1, P2, P3, P4, P5, P6, P7, and P8.

These symbols enable you to supply the names of files, variable values, and other information on the command line.

Passing Parameters to a Command Procedure

You can supply up to eight parameters when you execute a command procedure.

Format:

\$ @filename.com parameter1 parameter2 ... parameter8

- The *parameter* values are delimited by spaces.
- The values you specify are assigned to the local symbols P1 through P8, according to the position of the values on the command line.
- P1 through P8 are string symbols.

Example:

\$ @MONTHLY_REPORT APRIL NOPRINT

- The value of P1 is "APRIL" because that is the first parameter on the command line after the name of the command procedure.
- The value of P2 is "NOPRINT" because that is the second parameter



Note

Separate the values of the parameters P1 through P8 with spaces when invoking a command procedure interactively.

Separate the values of P1 through P8 with commas when specifying them to a batch command procedure by means of the /PARAMETERS qualifier to the SUBMIT command.

This may seem obvious, but point out that all eight parameters need not be specified to every command procedure.

Coding Without Parameters

The logical flow of a command procedure that requires three pieces of input might be as follows:

- The user invokes the command procedure.
`$ @MAILING_LIST.COM`
- The command procedure asks for the name of a mailing list to process, and stores the user's response in the local symbol `FILE_NAME`.
`$ INQUIRE FILE_NAME "Enter the file name"`
- The command procedure asks for the type of report to produce, and stores the response in the local symbol `REPORT_TYPE`.
`$ INQUIRE REPORT_TYPE "Enter the type of report"`
- The command procedure asks for the number of copies of the report to print, and stores the response in the local symbol `PRINT_NUMBER`.
`$ INQUIRE PRINT_NUMBER "Number of copies"`

This flow requires the user to answer these questions as the command procedure executes.

Coding with Parameters

A command procedure that requires three pieces of input might be coded to accept this information as parameters on the command line.

- Invoke the command procedure and supply three parameters.
`$ @MAILING_LIST.COM LISTA DAILY 5`
- When the command procedure needs the name of the mailing list, it tests whether P1 is equal to "". If P1 has a null value, the command procedure asks for the information; otherwise it uses the value of P1.
`$ IF P1 .EQS. "" THEN INQUIRE P1 "Enter the file name"`
`$ FILE_NAME = P1`
- When the command procedure needs the report type, it tests whether P2 is equal to "". If P2 does not have a null value, the command procedure asks for the information; otherwise it uses the value of P2.
`$ IF P2 .EQS. "" THEN INQUIRE P2 "Enter report type"`
`$ REPORT_TYPE = P2`
- When the command procedure needs the number of copies, it tests whether P3 is equal to "". If P3 does not have a null value, the command procedure asks for the information; otherwise it uses the value of P3.
`$ IF P3 .EQS. "" THEN INQUIRE P3 "Number of copies"`
`$ PRINT_NUMBER = P3`



Note

To pass parameters to a command procedure that is executed in batch mode, use the /PARAMETERS qualifier to the SUBMIT command.

Format:

SUBMIT/PARAMETERS=(parameter[,...]) filespec

Example: Displaying P1 Through P8

The following example shows how to use the DISPLAY_PARAMETERS command:

```
❶ $ TYPE DISPLAY_PARAMETERS.COM
$      ! DISPLAY_PARAMETERS.COM
❷ $ SHOW SYMBOL/LOCAL/ALL
$ EXIT
❸ $ @DISPLAY_PARAMETERS Argument_1 argument_2 v
❹ P1 = "ARGUMENT_1"
    P2 = "ARGUMENT_2"
    P3 = ""
    P4 = ""
    P5 = ""
    P6 = ""
    P7 = ""
    P8 = ""
❺ $ SHOW SYMBOL/LOCAL/ALL
$
```

Example Notes: Displaying P1 Through P8

Notes on the DISPLAY_PARAMETERS command include:

- ❶ Use the TYPE command to display the contents of the command procedure DISPLAY_PARAMETERS.COM. Note that this command procedure does not ask for, or use, any parameters.
- ❷ This line of the command procedure displays all local symbols for this command level.
- ❸ Invoke the command procedure, supplying two parameters. These are stored as P1 and P2 even though this command procedure does not use them.
- ❹ The output from DISPLAY_PARAMETERS.COM is a display of the values assigned to the parameters. Only two of the parameters have values, and these were converted to uppercase.
- ❺ At DCL level, the attempt to display all local symbols shows that P1 and P2 are no longer defined. They were local to the command procedure.

Preserving Case and Spacing in Parameters

To preserve the case and spacing of a string passed as a parameter to a command procedure, enclose the string in quotation marks.

Format:

\$ @filename.com "parameter1" ... "parameter8"

Example: Preserving Case and Spacing in Parameters

The following example show how to preserve case and spacing in parameters:

```
❶ $ TYPE DISPLAY_PARAMETERS.COM
$      ! DISPLAY_PARAMETERS.COM
❷ $ SHOW SYMBOL/LOCAL/ALL
$ EXIT
❸ $ @DISPLAY_PARAMETERS This is a STRING!
❹ P1 = "THIS"
    P2 = "IS"
    P3 = "A"
    P4 = "STRING"
    P5 = ""
    P6 = ""
    P7 = ""
    P8 = ""
❺ $ @DISPLAY_PARAMETERS "This is a STRING!"
❻ P1 = "This is a STRING!"
    P2 = ""
    P3 = ""
    P4 = ""
    P5 = ""
    P6 = ""
    P7 = ""
    P8 = ""
$
```

Example Notes: Preserving Case and Spacing in Parameters

Notes on how to preserve case and spacing in parameters include:

- ❶ Use the TYPE command to display the contents of the command procedure DISPLAY_PARAMETERS.COM.
- ❷ This line of the command procedure displays all local symbols.
- ❸ Invoke the command procedure, supplying a string containing spaces, upper- and lowercase characters, and an exclamation point (!).
- ❹ DISPLAY_PARAMETERS.COM interprets the string as four separate parameters. All strings have been converted to uppercase. Note that the exclamation point has been ignored because DCL interprets that as a comment character.

- ⑤ Invoke the command procedure again, enclosing the same string in quotation marks.
- ⑥ DISPLAY_PARAMETERS.COM interprets the string as one parameter. The case and spacing have been preserved, and the exclamation point has not been ignored because it is seen as part of the string.

Passing Integer Parameters

When integers are passed to a command procedure as parameters, DCL treats them as strings. If you want them to be treated as integers in calculations, you must include an integer value in the calculations.

Example: Passing Integer Parameters

The following example shows how to pass integer parameters:

```
① $ TYPE SUM.COM
   $!      SUM.COM
② $ SUM1 = P1 + P2
③ $ SHOW SYMBOL SUM1
④ $ SUM2 = 0 + P1 + P2
⑤ $ SHOW SYMBOL SUM2
   $ EXIT
⑥ $ @SUM 1 3
⑦ SUM1 = "13"
⑧ SUM2 = 4 Hex = 00000004 Octal = 00000000004
⑨ $ @SUM
   SUM1 = ""
   SUM2 = 0 Hex = 00000000 Octal = 00000000000
   $
```

Example Notes: Passing Integer Parameters

Notes on the Passing Integer Parameters example include:

- ① Display the contents of the command procedure SUM.COM.
- ② Create a symbol, SUM1, that represents the sum of the parameters P1 and P2. Because the parameters are string symbols, this is a string concatenation rather than an arithmetic sum.
- ③ Display the value of SUM1. The output from this command is shown in line ⑦.
- ④ Create a symbol, SUM2, that represents the sum of the integer 0 and the parameters P1 and P2. The presence of an integer in the expression will cause this to be an arithmetic sum. The integer 0 was used because it would not change the value of the expression.
- ⑤ Display the value of SUM2. The output from this command is shown in line ⑧.

- ⑥ Invoke the command procedure, supplying a value of 1 for P1 and a value of 3 for P2.
- ⑦ SUM1 is a string symbol, the result of concatenating "1" and "3".
- ⑧ SUM2 is a numeric symbol, the arithmetic sum of the integers 1 and 3.
- ⑨ Invoke the command procedure a second time, supplying no values for the parameters. Note that the results are a null string and the integer 0.

Controlling Flow of Execution

Overview

When you write a command procedure, you must be able to control what commands are executed under certain conditions. For example, you might test the value of a symbol, and depending on that value, select different tasks to carry out.

Using control commands you could also allow a user to enter parameters for the command procedure on the command line, or, not finding them, prompt the user to supply them.

Topics

In this section the following topics are discussed:

- Flow control commands
- Testing for conditions
- Transferring control within a command procedure
- Transferring control outside of a command procedure
- Iterative command procedures

Flow Control Commands

DCL Commands

The following DCL commands can be used to control the flow of execution in a command procedure.

IF - THEN	Tests the value of an expression and executes the specified command(s) if the result is true
GOTO	Transfers control to a different part of the procedure that is identified by a label
@	Temporarily transfers control to a command procedure, creating a new command level
CALL	Temporarily transfers control to a labeled subroutine within a command procedure, creating a new command level
GOSUB	Temporarily transfers control to a labeled subroutine within a command procedure without creating a new command level



Note

CALL and GOSUB are beyond the scope of this material, and are just included in this table for completeness.

Testing for Conditions

The IF-THEN command is used to test a condition and execute statements depending on the outcome of the test.

Using the IF-THEN Command

There are three ways to use the IF command:

1. Execute one statement if the conditional expression is true.
2. Execute a block of statements if the conditional expression is true.
3. Execute one statement or a block of statements if the conditional expression is true and execute a different statement or block of statements if the expression is false.

What Is a Conditional Expression?

A conditional expression can consist of one or more numeric constants, string literals, symbolic names, or lexical functions separated by logical, arithmetic, or string operators.

Examples:

```
COUNT .LT. 5
```

```
P1 .EQS. "PRINT"
```

```
F$LENGTH(DATA) .GT. 0
```

When Is a Conditional Expression True?

A conditional expression is true if it:

- Has an odd integer value
- Is a character string that begins with the letter Y, y, T, or t
- Is a character string that has an odd numeric value

Operators Used in Conditional Expressions

Comparison operators are used in conditional expressions to compare symbols or expressions.

Comparing Numeric Expressions

The following is a list of the numeric comparison operators.

This operator...	Has this meaning:
.EQ.	Equal to
.GE.	Greater than or equal to
.GT.	Greater than
.LE.	Less than or equal to
.LT.	Less than
.NE.	Not equal to

Comparing Strings

The following is a list of the string comparison operators. They all end in the letter S, for "string."

This operator...	Has this meaning:
.EQS.	String equal to
.GES.	String greater than or equal to
.GTS.	String greater than
.LES.	String less than or equal to
.LTS.	String less than
.NES.	String not equal to

Logical Operators

The following logical operators can also be used in expressions:

This operator...	Has this meaning:
.NOT.	Logically negates a number
.AND.	Combines two numbers with logical AND
.OR.	Combines two numbers with logical OR

Precedence of Operation

When operations are combined, they are performed in the following order of precedence:

Precedence	Operator
1	+ (positive number)
1	- (negative number)
2	*
2	/
3	+ (add two numbers or concatenate two strings)
3	- (subtract two numbers or subtract two strings)
4	.EQ. and .EQS.
4	.GE. and .GES.
4	.GT. and .GTS.
4	.LE. and .LES.
4	.LT. and .LTS.
4	.NE. and .NES.
5	.NOT.
6	.AND.
7	.OR.

Simple IF Command

The simplest form of the IF command executes a single statement when the result of a conditional expression is true.

Format:

\$ IF conditional-expression THEN command

- The *conditional expression* can consist of one or more numeric constants, string literals, symbolic names, or lexical functions separated by logical, arithmetic, or string operators.
- Do not place a \$ before the THEN command in a simple IF command.
- The *command* will be executed if the statement is true. This can be any valid DCL command, and must be on the same command line as the THEN command.

Example: IF Command

The following example uses the IF command:

```
$ IF COUNT .EQ. 0 THEN EXIT
```

If the conditional expression is true, the EXIT command will be executed.

Complex IF Command

To execute more than one statement when the conditional expression evaluates true, place the THEN statement on a separate command line and terminate the block of statements with the ENDIF statement.

Format:

```
$ IF conditional-expression
$ THEN
$   command
$   command
.
.
.
$ ENDIF
```

Example: Complex IF Command

The following examples uses the complex IF command:

```
$ IF COUNT .EQ. 0
$ THEN
$   WRITE SYS$OUTPUT "Procedure finished."
$   EXIT
$ ENDIF
.
.
.
```

Indenting the block of statements makes the command procedure easier to read and understand. It is not necessary for proper execution.

IF THEN ELSE Command

The IF THEN ELSE command executes one or more commands when the specified condition is true, and one or more commands when the condition is false.

Format:

```
$ IF conditional-expression  
$ THEN  
$ command  
$ command  
.  
.  
.  
$ ELSE  
$ command  
$ command  
.  
.  
.  
$ ENDI
```

Example: Using IF-THEN-ELSE

The following examples uses the IF-THEN-ELSE command:

```
$ INQUIRE NUM_COPY "How many copies do you want?"  
$ IF NUM_COPY .LT. 5  
$ THEN  
$ PRINT/QUEUE=OFFICE_PRINT MYFILE.TXT  
$ WRITE SYS$OUTPUT "Your copies are printing in the  
Office"  
$ ELSE  
$ PRINT/QUEUE=LAB_PRINT MYFILE.TXT  
$ WRITE SYS$OUTPUT "Your copies are printing in the Lab"  
$ ENDIF
```

Evaluation of Expressions in IF Commands

Expressions are automatically evaluated during the execution of an IF command.

- Character strings beginning with alphabetic characters that are not enclosed in quotation marks are assumed to be symbol names or lexical functions. The command language interpreter tries to replace these symbols with their current values.

The command interpreter does not execute an IF command that contains an undefined symbol. Instead, it issues a warning message and executes the next command in the procedure.

- Symbol substitution in expressions used in IF commands is not iterative. Each symbol is replaced only once.
- To force iterative substitution, precede a symbol name with an apostrophe.



Note

The *OpenVMS User's Manual* gives a summary of operators and details on how to specify expressions.

Transferring Control Within a Command Procedure

Flow of Execution

The normal flow of execution in a command procedure is sequential: the commands are executed, in order, until the end of the file is reached. The GOTO command can be used to cause a command procedure to branch forward or backward.

GOTO Command

Use the GOTO command to transfer control to a labeled line in a command procedure.

Format:

\$ GOTO label-name

Example: GOTO Command

The following example uses the GOTO command:

```
$ GOTO STEP_20
```

This command transfers control to the label STEP_20. Do not include a colon at the end of the label specifier in the GOTO statement.

Labels

Labels are used to identify blocks of code or routines that carry out a specific task within a command procedure.

Format:

\$ LABEL:

A label:

- Must appear as the first item on a line after the dollar sign (\$)
- May contain 1 through 255 alphanumeric characters including the underline (_) character
- Must be terminated with a colon (:)
- Cannot contain embedded space characters

Nonexistent Label

If a command procedure includes a GOTO statement that refers to a nonexistent label, the command procedure stops executing and exits.

The following error message displays:

```
%DCL-W-USGOTO, target of GOTO not found - check spelling and  
presence of label
```

Duplicate Labels

When the command interpreter encounters a label, it enters the label into a label table. If the label already exists in the table, the new definition replaces the existing one. Therefore, if you use duplicate labels, control is always given to the label most recently read by DCL.

Caution About Labels

The amount of space available in a process for labels is limited. If a command procedure uses many symbols and contains many labels, the command interpreter may run out of space and issue an error message.



Note

For more information about labels, see the GOTO section of the *OpenVMS DCL Dictionary*.

Example: Using the GOTO Command

The following example uses the GOTO command:

```
❶ $ TYPE GOTO.COM
$      ! GOTO.COM
❷ $ GOTO LUNCH    ! Unconditional transfer of control.
❸ $ SHOW USERS    ! We never execute this statement.
$
❹ $ LUNCH:        ! Target of GOTO command.
$ SHOW DEFAULT    ! DCL commands to show that control
$ SHOW TIME       ! was transferred.
$ EXIT           ! Terminate the command procedure.
❺ $ @GOTO
    WORK3: [SMITH]

18-DEC-1999 10:50:17

$
```

Example Notes: Using the GOTO Command

Notes on the Using the GOTO Command example include:

- ❶ Display the contents of the command procedure.
- ❷ Transfer control to the label LUNCH.
- ❸ This command will never be executed.
- ❹ This line is the target of the GOTO command in line (2).
- ❺ Execute the command procedure.

Example: Branching in a Command Procedure

The following example shows branching in a command procedure:

```
❶ $ TYPE GREETING.COM
$!   GREETING.COM
❷ $ DATE = F$TIME()
❸ $ TIME = F$EXTRACT(12,2,DATE)
❹ $ IF TIME .GE. 12 THEN GOTO AFTERNOON
$ MORNING:
$ WRITE SYS$OUTPUT "Good morning!"
❺ $ GOTO END_GREETING
❻ $ AFTERNOON:
$ WRITE SYS$OUTPUT "Good afternoon!"
$ END_GREETING:
$ WRITE SYS$OUTPUT DATE
$ EXIT
❼ $ @GREETING
Good afternoon!
18-DEC-1999 13:52:43.38
$
```

Example Notes: Branching in a Command Procedure

Notes on the Branching in a Command Procedure example include:

- ❶ Display the contents of the command procedure.
- ❷ Lexical function to get the date and time. The information is returned in the following format:
dd-mmm-yyyy hh:mm:ss.cc
- ❸ Lexical function to extract the hour. Skip the first 12 characters in the string and extract the next two characters.
- ❹ GOTO command included in THEN clause. If the hour is greater than or equal to 12, then branch to the label AFTERNOON.
- ❺ Unconditional GOTO statement.
- ❻ This line is the target of the GOTO command in line (4).
- ❼ Execute the command procedure.

Transferring Control Outside of a Command Procedure

Nesting Command Procedures

One command procedure can execute another command procedure. The maximum command level you can achieve by nesting command procedures in this way is 32, including the top-level command procedure.

@ Command

Use the @ command to execute another command procedure from within a command procedure. This creates a new command level.

Format:

\$ @filename.com

You can create a maximum of 32 command levels.

Availability of Symbols at Different Command Levels

When one command procedure invokes another command procedure:

- Global symbols defined in the original command procedure are available to the new procedure and can be used to pass values back to the original command procedure.
- Local symbols defined in the original command procedure are available to the new procedure but cannot be used to pass values back to the original command procedure.
- Local symbols defined at the new command level are available only to this level and any subsequent levels created from this level.

Iterative Command Procedures

Iteration

Iteration is the repetition of a group of instructions. To write iterative procedures, use a looping structure.

Example: Iterative Command Procedure

The following example shows how to use the ITERATE command:

```
❶ $ TYPE ITERATE.COM
$      ! ITERATE.COM
$ COUNT = 0      ! Initialize counter for loop.
❷ $ LOOP:      ! Label for looping structure.
$ COUNT = COUNT + 1 ! Increment the counter.
$ WRITE SYS$OUTPUT "COUNT=", count ! Display value of
counter.
❸ $ IF COUNT .LT. 5 THEN GOTO LOOP ! Iterate the loop.
$ WRITE SYS$OUTPUT "Done"      ! Loop is finished.
$ EXIT
❹ $ @ITERATE.COM
COUNT = 1 Hex = 00000001 Octal = 00000000001
COUNT = 2 Hex = 00000002 Octal = 00000000002
COUNT = 3 Hex = 00000003 Octal = 00000000003
COUNT = 4 Hex = 00000004 Octal = 00000000004
COUNT = 5 Hex = 00000005 Octal = 00000000005
Done
$
```

Example Notes: Iterative Command Procedure

Notes on the Iterative Command Procedure example include:

- ❶ Display the contents of the command procedure.
- ❷ This is the label for the looping structure.
- ❸ This statement causes the iteration.
- ❹ Execute the command procedure.

Redirecting Input and Output

Overview

You can redefine the logical names `SYSS$OUTPUT` and `SYSS$INPUT` to display and receive information in command procedures.

Topics

In this section the following topics are discussed:

- Redefining `SYSS$OUTPUT`
- Redefining `SYSS$INPUT`

Redefining SYS\$OUTPUT

Redefining SYS\$OUTPUT

You can redefine SYS\$OUTPUT to capture output for later reference. You might want to keep:

- A directory listing for future reference
- A record of output from a program
- A list of your mail messages
- A copy of an error message

Directing Output to a File

You can redirect terminal output to a file by using the ASSIGN or DEFINE commands to give a new value to the logical name associated with your terminal.

Format:

\$ ASSIGN *output_file* SYS\$OUTPUT

\$ DEFINE SYS\$OUTPUT *output_file*

Sample commands:

\$ ASSIGN STATS.DAT SYS\$OUTPUT

\$ DEFINE SYS\$OUTPUT STATS.DAT

This assignment lasts until you log out, assign SYS\$OUTPUT another value, or enter the DEASSIGN command. Deassigning SYS\$OUTPUT does not remove the logical name, it just removes the equivalence you have given the logical name. The equivalence assigned by the system remains.

Example: Directing Output to a File

The following example shows how to direct output to a file. The file can later be edited, sent to a printer, mailed to another user, and so on.

```
❶ $ SHOW DEFAULT
    WORK3: [USER6]
❷ $ ASSIGN DEFAULT.TXT SYS$OUTPUT
❸ $ SHOW DEFAULT
❹ $ DEASSIGN SYS$OUTPUT
❺ $ TYPE DEFAULT.TXT
    WORK3: [USER6]
$
```

Example Notes: Directing Output to a File

Notes on the Directing Output to a File example include:

- ❶ Output from this command is displayed at the terminal.
- ❷ Give a new definition to the logical name SYS\$OUTPUT, directing output to the file called DEFAULT.TXT.

The ASSIGN command was used in this example. To use the DEFINE command, reverse the order of the parameters:

```
$ DEFINE SYS$OUTPUT DEFAULT.TXT
```

- ❸ Now when you enter the SHOW DEFAULT command, the output does not appear at the terminal.
- ❹ Remove your definition of SYS\$OUTPUT so any further output will be displayed at your terminal. The definition assigned by the system remains.
- ❺ DEFAULT.TXT contains the output from the SHOW DEFAULT command.

Temporarily Redefining SYS\$OUTPUT

You can make a temporary assignment by adding the /USER qualifier to the ASSIGN or DEFINE command:

```
$ ASSIGN/USER_MODE STATS.DAT SYS$OUTPUT
$ DEFINE/USER_MODE SYS$OUTPUT STATS.DAT
```

This assignment will last only during the execution of the next image (program).

Example: Temporarily Directing Output to a File

The following example show how to direct output to a file:

```
❶ $ ASSIGN/USER_MODE USER.TXT SYS$OUTPUT
❷ $ SHOW USERS
❸ $ TYPE USER.TXT
❹ OpenVMS User Processes at 08-NOV-1999 11:19:36.41
   Total number of users = 5, number of processes = 7

Username Node   Interactive Subprocess Batch
.
.
.
$
```

Example Notes: Temporarily Directing Output to a File

Notes on the Directing Output to a File example include:

- ❶ Temporarily reassign the logical name SYS\$OUTPUT to the file USER.TXT.
- ❷ Enter the SHOW USERS command. There will be no response on your terminal because output for this command was directed to the file USER.TXT.
- ❸ TYPE the file USER.TXT to see the output of the command.
- ❹ This time, output **does** occur at the terminal because the reassignment of the logical name was only for one program execution.

Redefining SYS\$INPUT

You can redefine SYS\$INPUT to allow a command procedure to read input from the terminal or another file. Normally, the system will only read input from the command file itself; but by redefining it, you can force the system to read instructions from another file or from the terminal.

Editing a File from a Command Procedure

Editors take their input from SYS\$INPUT. When a command procedure is executing, SYS\$INPUT is the command procedure.

To edit a file from a command procedure:

```
$ ASSIGN/USER_MODE SYS$COMMAND SYS$INPUT
$ EDIT MYFILE.DAT
```

- SYS\$COMMAND always refers to the terminal, the initial input stream when you logged in.
- The /USER_MODE qualifier redefines SYS\$INPUT only for the duration of the next image (the editor).
- You can edit interactively, then exit from the editor.
- When the editor image exits, SYS\$INPUT resumes its default value, the command procedure file.

Running an Interactive Program from a Command Procedure

Images take their input from SYS\$INPUT. When a command procedure is executing, SYS\$INPUT is the command procedure.

To run an interactive program from a command procedure, include the following lines in the command procedure:

```
$ ASSIGN/USER_MODE SYS$COMMAND SYS$INPUT
$ RUN MONTHLY_REPORT
```

- While the image is executing, it takes its input from the terminal (SYS\$COMMAND).
- When the image exits, control returns to the command procedure.

Using Lexical Functions

Overview

Lexical functions are command language constructs used to obtain and manipulate information about the current process, the system, and arithmetic and string expressions. Some lexical functions return information that you can also get by using DCL commands. The information is returned as a symbol value that can be manipulated in performing tasks.

Topics

In this section the following topics are discussed:

- Lexical functions
- Manipulating character strings
- Obtaining and manipulating data

Lexical Functions

What Are Lexical Functions?

Lexical functions are command language constructs that return integer values or character strings depending on their particular function.

Lexical functions can be used in any context in which you normally use symbols or expressions. They are most commonly used in command procedures, although they can be used interactively.

Some examples of information that can be obtained by using a lexical function are the:

- System time
- Current default directory
- Length of a string

Format of Lexical Functions

All lexical functions begin with F\$, followed by the function name.

Format:

F\$function-name([argument,...])

- The *function-name* is the name of the lexical function. This name usually gives some indication of what the function does.
- The *arguments* for the function, if any, are enclosed in parentheses.
- Multiple arguments are separated by commas.
- Optional arguments, when omitted, are still represented by commas.

Arguments

Arguments for a lexical function can be:

- Integer or character strings as a literal value
`WHAT = F$EXTRACT(0,3,"MAILMAN")`
- Symbols
`HOWLONG = F$LENGTH(X)`
- Keywords
`WHERE = F$ENVIRONMENT("DEFAULT")`
- Null arguments
`WHEN = F$TIME()`

Parentheses are required even for null values.

Manipulating Character Strings

Lexical functions can be used to obtain and manipulate character strings. Some of the operations you can perform are:

- Edit a string
- Extract specific parts of a character string
- Return the length of a string
- Locate a substring within a string and return the offset position
- Return character strings in a specified format
- Return the integer equivalent of a specified string

The following pages give examples of lexical functions that are commonly used to manipulate character strings.

F\$EDIT

Use the F\$EDIT lexical function to edit a string expression.

Format:

F\$EDIT(string,edit-list)

- The *string* argument contains the string expression to be edited.
This argument can be a group of literal characters enclosed in quotation marks, or a symbol that contains a string value.

- Values you can supply for the *edit-list* argument include:

COLLAPSE	Removes all spaces and tabs from the string
COMPRESS	Replaces multiple spaces and tabs with a single space
LOWERCASE	Makes the string lowercase
TRIM	Removes leading and trailing spaces and tabs from the string
UNCOMMENT	Removes comments from the string
UPCASE	Makes the string uppercase

- Multiple edit-list items can be combined using commas. This argument can be specified as a group of literal characters enclosed in quotation marks, or as a symbol representing a string value.

Example: F\$EDIT

The following example shows how to use F\$EDIT:

```
$ LONG = " this symbol is in LowerCaSe and has some spaces"
$ SHORT = F$EDIT(LONG,"COMPRESS,UPCASE")
$ SHOW SYMBOL SHORT
SHORT = " THIS SYMBOL IS IN LOWERCASE AND HAS SOME SPACES"
$
```



Note

- Quoted sections of the string are not edited.
 - In the example, point out that multiple edit-list items can be combined with commas.
-

F\$LENGTH

Use the F\$LENGTH lexical function to determine the length of a string.

Format:

F\$LENGTH(string)

- Specify the *string* argument as a character string expression (in quotation marks), or as a symbol name.

Example: F\$LENGTH

The following example shows how to use F\$LENGTH:

```
$ LEN = F$LENGTH("HOW LONG IS THIS STRING?")
$ SHOW SYMBOL LEN
LEN = 24 Hex = 00000018 Octal = 00000000030
$
```



Note

Specify the string argument as a character string expression (in quotes) or as a symbol name.

F\$LOCATE

Use the F\$LOCATE lexical function to locate a specified substring in a string.

Format:

F\$LOCATE(substring,string)

- The returned value is the offset of the first occurrence of the *substring* in the specified *string*.

If the substring is found, this value ranges from 0 to 1 less than the length of the searched string.

If the substring is not found, F\$LOCATE returns the length of the searched string.

- Specify the *substring* and *string* arguments as a character string expression (in quotation marks), or as a symbol name.

Example: F\$LOCATE

The following example shows how to use F\$LOCATE

```
$ LONG_STRING = "This is a long string. We want the word  
long"  
$ FIND = F$LOCATE("long",LONG_STRING)  
$ SHOW SYMBOL FIND  
FIND = 10 Hex = 0000000A Octal = 00000000012  
$ FIND = F$LOCATE("short",LONG_STRING)  
$ SHOW SYMBOL FIND  
FIND = 45 Hex = 0000002D Octal = 00000000055  
$
```

F\$EXTRACT

Use the F\$EXTRACT lexical function to extract a substring from a character string expression.

Format:

F\$EXTRACT(offset,length,string)

- The *offset* argument is the position, relative to the beginning of the string, of the string to be extracted. If the offset is 0, the extraction starts with the first character of the string.
- The *length* argument specifies the number of characters to be extracted.
- The *string* argument is the source string from which the substring is to be extracted.

Example: F\$EXTRACT

The following example shows how to use F\$EXTRACT

```
$ FULL_NAME = "FRED CLOGGS"
$ FIRST = F$EXTRACT(0,4,FULL_NAME)
$ SHOW SYMBOL FIRST
  FIRST = "FRED"
$
```



Note

If the offset is 0, the extraction starts with the first character of the string.

Obtaining and Manipulating Data

Lexical functions can be used to get information about:

- Your process
- The system
- Files and devices
- Logical names
- Data types

F\$DIRECTORY

Use the F\$DIRECTORY lexical function to obtain the current default directory as a character string. (This does not return the associated device.)

Format:

F\$DIRECTORY()

Example: F\$DIRECTORY

The following example shows how to use F\$DIRECTORY

```
$ SHOW DEFAULT
USER_DISK:[USER7.SUB1]
$
$ CURR_DIR = F$DIRECTORY( )
$ SHOW SYMBOL CURR_DIR
CURR_DIR = "[USER7.SUB1]"
$
$ SET DEFAULT SYS$LOGIN
$ SHOW DEFAULT
USER_DISK:[USER7]
$
$ SET DEFAULT 'CURR_DIR'
$ SHOW DEFAULT
USER_DISK:[USER7.SUB1]
$
```



Note

Points to make about the F\$DIRECTORY example:

- Capture the current default directory in the symbol CURR_DIR. F\$DIRECTORY does not take any arguments, but you must still supply parentheses.
 - You must force symbol substitution using apostrophes when you set the default back to CURR_DIR because SET DEFAULT does not expect to be given a symbol.
-

F\$MODE

Use the F\$MODE lexical function to obtain a character string indicating the mode in which a process is running (for example, INTERACTIVE, NETWORK, BATCH, or OTHER).

Format:

F\$MODE()

- This lexical function accepts no arguments.
- The returned string always contains uppercase letters.

Example: F\$MODE

The following example shows how to use F\$MODE

```
$ MODE = F$MODE( )
$ SHOW SYMBOL MODE
MODE = "INTERACTIVE"
$
```



Note

Point out that this is a useful lexical function to include in LOGIN.COM. There may be commands in that file that are only appropriate to execute interactively. The command procedure can skip these commands if the value of F\$MODE is not "INTERACTIVE".

F\$TIME

Use the F\$TIME lexical function to obtain the current date and time string.

Format:

F\$TIME()

- This lexical function takes no arguments, but parentheses must still be used.
- The returned string has the following fixed, 23-character format:
dd-mmm-yyyy hh:mm:ss.cc
- When the current day of the month is any of the values from 1 through 9, the first character in the returned string is a blank character.

Example: F\$TIME

The following example shows how to use F\$TIME

```
$ NOW = F$TIME( )
$
$ SHOW SYMBOL NOW
NOW = " 18-DEC-1999 11:12:56.81"
$
```



Note

The time portion of the returned string is always in character position 13, at an offset of 12 characters from the beginning of the string.

Note that the time in NOW stays at the time you created the symbol. If you enter the SHOW SYMBOL NOW command 10 minutes later, it will not display a new time.

F\$CVTIME

Use the F\$CVTIME lexical function to obtain information about absolute, combination, or delta time strings.

Format:

F\$CVTIME([input-time][,output-time-format][,output-time-field])

Some of the arguments for this lexical function include:

- *Output-time-format*

ABSOLUTE	The requested information should be returned in absolute time format.
COMPARISON	The requested information should be returned in the form "yyyy-mm-dd hh:mm:ss.cc". (This is the default argument.)
DELTA	The requested information should be returned in delta format. If you specify delta as the output time argument, you must also provide a delta time specification for the input time argument.

- *Output-time-field*

DATE	The date field is returned.
DATETIME	The entire date and time string is returned. (This is the default argument.)
DAY	The day field is returned.
HOUR	The hour field is returned.
MONTH	The month field is returned. You cannot specify MONTH if you also specify a delta input time and a delta output time argument.
TIME	The time field is returned.
WEEKDAY	The weekday that corresponds with the input time argument is returned. You cannot specify WEEKDAY if you also specify a delta input time and a delta output time argument.
YEAR	The year field is returned. You cannot specify YEAR if you also specify a delta input time and a delta output time argument.

Example: F\$CVTIME

The following example shows how to use F\$CVTIME

```
$ DAY = F$CVTIME("TODAY", , "WEEKDAY")
$ SHOW SYMBOL DAY
DAY = "Thursday"
$
$ NEXT = F$CVTIME("TOMORROW", , "WEEKDAY")
$ SHOW SYMBOL NEXT
NEXT = "Friday"
$
```



Note

- You can omit optional arguments that can be used to the right of the last argument you specify:

```
$ TIME = F$TIME()
$ SHOW SYMBOL TIME
TIME = "18-DEC-1999 14:13:36.83"
$ TIME = F$CVTIME(TIME)
$ SHOW SYMBOL TIME
TIME = "1999-09-18 14:13:36.83"
```

- If the input-time argument is omitted or is a null string (""), the current system data and time, in absolute format, is used.
 - If the input-time argument is a delta time and the output-time-format argument is DELTA, you cannot specify MONTH, WEEKDAY, or YEAR for the output-field.
 - When the weekday is returned, the first letter is in uppercase and the rest are in lowercase.
 - Other values for the output-time-field:
HUNDREDTH The hundredth of a second field is returned
MINUTE The minute field is returned.
SECOND The second field is returned.
-

F\$GETSYI

Use the F\$GETSYI lexical function to obtain status and identification information about your system or a node in your cluster.

Format:

F\$GETSYI(item,[node])

Some of the item codes are:

CLUSTER_MEMBER	"TRUE" if the node is currently in the cluster.
NODENAME	Returns the node name (string).
NODE_SWTYPE	Returns the type of operating system software used by the specified node (string).
NODE_SWVERS	Returns the software version of the specified node (string).
NODE_HWTYPE	Type of hardware.

Example: F\$GETSYI

The following example shows how to use F\$GETSYI.

Suppose, in your startup command procedure, you want to start certain layered products only on certain nodes of a cluster. Use code like the following:

```
$ NODE = F$GETSYI("NODENAME")
.
.
.
$ @SYS$STARTUP:PROD1_STARTUP
$ IF NODE .EQS. "BARNUM" THEN
$   @SYS$STARTUP:PROD2_STARTUP
$ ENDIF
$ IF NODE .EQS. "LION" .OR. NODE .EQS. "BEAR" THEN
$   @SYS$STARTUP:PROD3_STARTUP
$   @SYS$STARTUP:PROD4_STARTUP
$ ENDIF
.
.
.
```

Only a few items have been listed here. For a complete listing, refer students to the *OpenVMS DCL Dictionary*.

Returned values are either integers or character strings, depending upon the item you request.

A logical question is how to find out what nodes are in the cluster. If it comes up, mention F\$CSID.

Writing Command Procedures That Perform I/O

Overview

As a System Manager you can write command procedures that access and perform actions on specified files.

Topics

In this section the following topics are discussed:

- File access steps
- Opening a file
- Writing to a file
- Reading from a file
- Closing a file
- Handling file I/O errors

File Access Steps

The basic steps in reading from, and writing to, files from a command procedure are:

1. Use the OPEN command to open the file.
2. Use the READ or WRITE commands to read or write records.
 - a. Read a record.
 - b. Perform some task on the record.
 - c. Write the record, then read a new record until the end of the file.
3. Use the CLOSE command to close the file.

Unless you explicitly close the file, it remains open until you log out.

Opening a File

A file can be opened for:

- Read access (the default)
- Write access
- Both read and write access

The OPEN command assigns a logical name to the file. Subsequent READ, WRITE and CLOSE commands use this logical name to refer to the file. Do not use the same logical name when you open different files.

Files opened by a command procedure are **not** closed automatically when the command procedure ends. The command procedure must **explicitly close** files.

OPEN Command

The OPEN command tells the OpenVMS operating system to prepare the file for I/O. This involves setting up data structures and fetching file characteristics.

Format:

\$ OPEN[/qualifier(s)] logical-name[:] file-specification

- *Qualifiers* to the OPEN command determine the access to be allowed and position the record pointer.
- Once the file has been opened, the command procedure will access it by specifying the *logical-name* used in the OPEN command. Do not use the same logical name when you open different files.
- When using the OPEN command in a command procedure, use the complete *file-specification* so that the user need not set default to the directory when running the command procedure. (Logical names are helpful here.)
 - Wildcard characters are not permitted in the file specification.
 - The file type defaults to DAT.

OPEN/READ Command

Use the OPEN/READ command to open an existing file. This is the default if you open a file without specifying any qualifiers.

- If the file is found, the record pointer will be positioned at the beginning of the file. Access to the file will be limited to reading records.
- If the specified file is not found, an error will be generated.

Example: OPEN/READ

The following example shows how to use the OPEN/READ command.

```
$ OPEN/READ INFILE USER8:[SMITH]SAMPLE.DAT
```

OPEN/WRITE Command

Use the OPEN/WRITE command to open a file for writing records.

- If the specified file is found, a new version of the file will be created. This new version will be opened and empty.
- If the file is not found, a new file will be created.

Example: OPEN/WRITE

The following example shows how to use the OPEN/WRITE command.

```
$ OPEN/WRITE OUTFILE USER8:[SMITH]OUTPUT.DAT
```

OPEN/WRITE /READ Command

If you combine the /WRITE and /READ qualifiers when you open an existing file, the record pointer will be placed at the beginning of the file so records can be read and then changed. When you use this method of updating a file, you can replace the record just read with a new record of the same size.

Example: OPEN/WRITE/READ

The following example shows how to use the OPEN/WRITE/READ command.

```
$ OPEN/WRITE/READ IN_OUT_FILE USER8:[SMITH]PERMANENT.DAT
```

OPEN/APPEND Command

Use the OPEN/APPEND command to add data to the end of an existing file.

This command opens the file for writing and positions the record pointer at the end of the file. This allows the addition of records to the file.

Example: OPEN/APPEND

The following example shows how to use the OPEN/APPEND command.

```
$ OPEN/APPEND OUTFILE USER8:[SMITH]PERMANENT.DAT
```

The /WRITE and /APPEND qualifiers are mutually exclusive and cannot be used together.

OPEN/ERROR Command

Use the qualifier /ERROR=label to transfer control to the specified label if an error occurs when you attempt to open a file.

Example: OPEN/ERROR

The following examples show how to use the OPEN/ERROR command.

```
$ OPEN/READ/ERROR=ERROR_HANDLER INFILE USER8:[SMITH]IN.DAT  
$ OPEN/WRITE/ERROR=ERROR_HANDLER OUTFILE  
USER8:[SMITH]OUT.DAT
```

Writing to a File

WRITE Command

Use the WRITE command to write a single record to a file.

Format:

\$ WRITE[/qualifier(s)] logical-name[:] expression

- *Qualifiers* can be combined.
- Use the *logical-name* that was assigned in the OPEN command.
- Specify the data to be written as a symbol or a character string expression. WRITE assumes a symbol. A list of expressions separated by commas will be concatenated into one record.

WRITE/ERROR Command

Use the WRITE/ERROR=label command to transfer control to the specified label if an error occurs during the write operation.

\$ WRITE/ERROR=ERROR_HANDLER OUTFILE DATA_RECORD

WRITE/UPDATE Command

Use the WRITE/UPDATE command to replace the last record read with the record specified in this WRITE command.

\$ WRITE/UPDATE OUTFILE DATA_RECORD

- Use this command only after a READ command.
- With sequential files, you must replace a record with another record of the same size when you use WRITE/UPDATE.

Example: Using WRITE in a Command Procedure

The following example shows how to use the WRITE command:

```
❶ $ OPEN/WRITE OUT DATA.NEW
   $ WRITE_LOOP:
❷ $ INQUIRE DATA

❸ $ IF DATA .EQS. "END" THEN GOTO QUIT

❹ $ WRITE/SYMBOL OUT DATA
   $ GOTO WRITE_LOOP
   $
   $ QUIT:
   $ CLOSE OUT
   $ EXIT
```

Example Notes: Using WRITE in a Command Procedure

Notes on the Using WRITE in a Command Procedure example include:

- ❶ Open the file for write access. Assign the logical name OUT to the file.
- ❷ Request input from the terminal, and store it in the local symbol DATA.
- ❸ The user must type "END" to cause the command procedure to terminate. (Any combination of upper- and lowercase will work because INQUIRE converts the input to uppercase.)
- ❹ Write the contents of the symbol DATA to the file. /SYMBOL puts the value into a 2048-byte buffer instead of a 1424-byte buffer. The use of the /SYMBOL qualifier is to write a very large record.

Reading from a File

READ Command

Use the READ command to read a record and assign its contents to a symbol.

Format:

\$ READ[/qualifier(s)] logical_name[:] symbol-name

- *Qualifiers* can be combined.
- The *label* is the label to which the program should transfer control when the end of the file is reached.
- The *logical_name* parameter is the logical name assigned to the file when it was opened with the OPEN command.
- The *symbol-name* parameter is a local symbol that will be equated to the value read from the file.

After each record is read from the specified file, the record pointer is positioned at the next record in the file. The maximum record size for a single READ is 2048 bytes.

The READ command accepts data exactly as it is found in the file. It does **not** convert characters to uppercase, remove extra spaces and tabs, or remove quotation marks.

READ/ERROR Command

Use the READ/ERROR=label command to transfer control to the specified label if an error occurs during the read operation or when the end of the file is reached.

Example:

```
$ READ/ERROR=ERROR_HANDLER INFILE DATA_RECORD
```

READ /END_OF_FILE Command

Use the READ/END_OF_FILE=label command to transfer control to the specified label if the end of the file is encountered during the read operation.

Example:

```
$ READ/END_OF_FILE=DONE INFILE DATA_RECORD
```

Example: Using a Command Procedure to Read a File

This segment of a command procedure opens a file for read access.

```
❶ $ OPEN/ERROR=ERROR_OPEN/READ INFILE SAMPLE.DAT
$ READ_LOOP:
$ READ/END_OF_FILE=ENDIT INFILE RECORD
.
.
.
$ GOTO READ_LOOP
❷ $ ENDIT:
❸ $ CLOSE INFILE
$ EXIT
❹ $ ERROR_OPEN:
.
.
.
```

Example Notes: Using a Command Procedure to Read a File

Notes on the Using a Command Procedure to Read a File example include:

- ❶ Open the file for read access.
- ❷ Read a record from the file. If the end-of-file is encountered, control is transferred to the label ENDIT.
- ❸ Close the file.
- ❹ Go to this label if an error is encountered while trying to open the file.

Closing a File

The command interpreter does not automatically close files.

- Files that are open for reading or writing at the command level remain open until they are explicitly closed, or until the process that has them open terminates.
- If a command procedure that opens a file terminates without closing it, the file remains open. Later attempts to access this file will generate errors.

CLOSE Command

Use the CLOSE command to close a file and deassign the associated logical name.

Format:

\$ CLOSE[/qualifier(s)] logical-name[:]

- *Qualifiers* can be combined.
- Refer to the file by the *logical-name* that was specified when the file was opened.

Example: CLOSE

The following example shows how to use the CLOSE command:

```
$ OPEN INFILE USER8:[SMITH]SAMPLE.DAT
.
.
.
$ CLOSE INFILE
```

CLOSE/ERROR Command

Use the CLOSE/ERROR=label command to transfer control to the specified label if an error occurs when attempting to close the file.

```
$ CLOSE/ERROR=CLOSE_ERROR INFILE
```

Example: Using OPEN and CLOSE

The following example shows how to use the OPEN and CLOSE commands:

```
❶ $ OPEN/WRITE OUTPUT FILEX.DAT
$ COUNT = 0
$ WRITE_LOOP:
$   COUNT = COUNT + 1
$   WRITE OUTPUT "Count is ",COUNT
$   IF COUNT .EQ. 10 THEN GOTO ENDIT
.
.
.
$   GOTO WRITE_LOOP
$ ENDIT:
❷ $   CLOSE OUTPUT
$   EXIT
```

Example Notes: Using OPEN and CLOSE

Notes on the Using OPEN and CLOSE example include:

- ❶ Open the file called FILEX.DAT for write access. Assign the logical name OUTPUT to the file. If an error occurs in opening the file, control will be transferred to the label ERROR_OPEN.
- ❷ Close the file, specifying the logical name, not the file specification. Closing a file deassigns the logical name that was created with the OPEN statement.

Handling File I/O Errors

A command procedure can anticipate and handle errors that may occur during file I/O. When an error occurs, control is transferred to the label specified in the command that generates the error.

/ERROR Qualifier

Use the `/ERROR=label` qualifier with the `OPEN`, `READ`, `WRITE`, and `CLOSE` commands to handle errors encountered while working with files.

Format:

\$ OPEN/ERROR=label logical-name file-name

\$ READ/ERROR=label logical-name symbol-name

\$ WRITE/ERROR=label logical-name expression

\$ CLOSE/ERROR=label logical-name

If an error occurs in one of these operations, control passes to the specified *label*. This allows you to write special error handling routines for file problems.

READ /END_OF_FILE Command

Use the `/END_OF_FILE=label` qualifier with the `READ` command to handle the situation in which the end of the file is encountered during a read operation.

Format:

\$ READ/END_OF_FILE=label logical-name symbol-name

If the end of the file is encountered in the read operation, control passes to the specified *label*.

The `/ERROR` qualifier can be combined with the `/END_OF_FILE` qualifier. In the absence of the `/END_OF_FILE` qualifier, `/ERROR` handles the end of file condition.

Example: Handling File I/O Errors

The following example show how to work with File I/O errors:

```
$          ! FILE_ERROR.COM
$          ! Handle file I/O errors
❶ $ OPEN/READ/ERROR=NO_FILE CALLS LOGFILES:CALL.LOG
$ READ_LOOP:
❷ $READ/END_OF_FILE=DONE CALLS DATA_LINE
$WRITE SYS$OUTPUT DATA_LINE
$GOTO READ_LOOP
❸ $ NO_FILE:
$ WRITE SYS$OUTPUT "Call log not found."
$ EXIT
❹ $ DONE:
$ CLOSE CALLS
$ EXIT
```

Example Notes: Handling File I/O Errors

Notes on the Handling File I/O Errors example include:

- ❶ Attempt to open the file, assuming that it exists. If the file does not exist, transfer control to the label NO_FILE.
- ❷ Read a record from the file. If the end of the file is encountered, transfer control to the label DONE.
- ❸ Control is transferred to the label NO_FILE if the OPEN command fails. A message is displayed, and the command procedure exits.
- ❹ When the end of the file is encountered in a read operation, control is transferred to this label. The command procedure closes the file, and exits.

Error Conditions and Ctrl/Y

Overview

It is important to detect errors in command procedures and to handle errors so that they do not escape detection.

There are two basic approaches to handling errors:

- Reactive error handling, in which you accept the system default action or specify your own action after an error is detected.
- Preventive error handling, in which you anticipate certain errors and override the default error handling with your own error handlers.

Topics

In this section the following topics are discussed:

- Detecting errors in command procedures
- Handling errors in command procedures
- Handling Ctrl/Y interrupts

Detecting Errors in Command Procedures

Condition Code

When each DCL command in a command procedure completes execution, the command interpreter saves a condition code that tells how the command terminated. The code indicates that the command has performed one of the following actions:

- Executed successfully
- Terminated with a warning
- Terminated with an informational message
- Terminated with an error
- Terminated with a severe or fatal error

How Condition Codes Are Returned

The system places condition codes in the global symbol `$STATUS`.

The severity of the condition code is represented in the three low-order bits of `$STATUS`. This severity level is also represented by the global symbol `$SEVERITY`.

Format of \$STATUS

The global symbol \$STATUS has the following format:

- Bits 0–2 contain the severity level of the message and are referenced by the global symbol \$SEVERITY
- Bits 3–15 contain the message number.
- Bits 16–27 contain the number associated with the facility that generated the message.
- Bits 28–31 contain internal control flags.

Values for \$SEVERITY

The following table shows the meanings of the values of \$SEVERITY.

This value...	Has this meaning...	And causes the command procedure to...
0	Warning (W)	Continue
1	Success (S)	Continue
2	Error (E)	Exit
3	Information (I)	Continue
4	Severe or fatal error (F)	Exit

Note that even values of \$SEVERITY indicate an error condition, while odd values indicate that no error has occurred.

Handling Errors in Command Procedures

Handling Errors

There are two ways you can handle errors in a command procedure:

- Use the ON command to specify the action to be taken when an error condition occurs in a command procedure.
- Use the SET NOON command to specify that no action is to be taken when an error occurs. (The SET ON command restores default error handling.)

ON Command

Use the ON command to specify a command to be executed when an error condition is encountered in a command procedure. The command will be executed when an error **equal to** or **greater than** the specified error condition occurs.

Format:

\$ ON error-condition THEN command

- There are three choices for *error-condition*.

WARNING	Warning, error, or severe errors cause <i>command</i> to execute.
ERROR	Error or severe errors cause <i>command</i> to execute.
SEVERE	Severe errors cause <i>command</i> to execute.

- *Command* specifies the command to be executed and can be any valid DCL command.

Example: ON

The following example shows how to use the ON command:

```
$ ON WARNING THEN GOTO ERROR_HANDLER
```

Any error causes control to be transferred to the label ERROR_HANDLER.

Default ON Action

The default error handling is ON ERROR THEN EXIT. The command procedure continues executing when a warning occurs, and executes an EXIT command when an error or severe error occurs.

There is an exception to this default action. When you specify a nonexistent label in a GOTO command, the command procedure issues a warning message and exits.

Guidelines for Using the ON Command

Keep the following points in mind when you use the ON command in a command procedure.

- An ON error-condition command is executed only once.
After the specified action is taken, the default ON action is reset. If you want your error handling to remain in effect, you must reissue the ON command after handling the error.
- Make sure your error handler is protected by an EXIT or GOTO command so the error handling statements are not executed accidentally.
- Only one ON command is in effect at a time. If you issue more than one ON command, the most recently issued command determines the error handling.
- The action specified by the ON command applies **only** within the command procedure in which the command is executed.

An ON command executed at any command procedure level does not affect the error condition handling of procedures at any other level.

If you execute an ON command in a command procedure that calls another procedure, the ON command action does not apply to the nested procedure.

- Although use of the ON command controls the action taken in the event of an error condition, the system still displays error messages. Use the SET MESSAGE command to disable these messages.

\$ SET MESSAGE/NOFACILITY/NOSEVERITY/NOIDENTIFICATION/NOTEXT

Example: Using the ON Command

The following example shows how the ON command is used:

```
$! CHECK_FILES.COM
$! Check for presence of necessary files.
$
$ STEP_1:          ! Check for master file.
❶ $  ON WARNING THEN GOTO ERROR_HANDLER_1
$  DIRECTORY SALES:MASTER_FILE.DAT
$  MESSAGE_1 = "Master file ready" ! If this command
$          ! executes, master file is present.
❷ $  GOTO STEP_2    ! Do not execute error handler.
❸ $  ERROR_HANDLER_1: ! First error handler.
$  MESSAGE_1 = "MASTER_FILE.DAT not found"
$  STEP_2:          ! Check for transaction file.
❹ $  ON WARNING THEN GOTO ERROR_HANDLER_2
$  DIRECTORY SALES:TRANSACTION.DAT
$  MESSAGE_2 = "Transaction file ready" ! If this command
$          ! executes, transaction file is present.
❺ $  GOTO END      ! Do not execute error handler.
$  ERROR_HANDLER_2: ! Second error handler.
$  MESSAGE_2 = "TRANSACTION.DAT not found"
$  END:            ! Display messages and terminate.
$  WRITE SYS$OUTPUT MESSAGE_1

$  WRITE SYS$OUTPUT MESSAGE_2
$  EXIT
```

Example Notes: Using the ON Command

Notes on the Using the ON Command example include:

- ❶ Establish the first error handler. Any error causes control to be transferred to the label ERROR_HANDLER_1.
- ❷ Prevent accidental execution of the first error handler.
- ❸ This is the first error handler. The command(s) at this label execute only if an error has occurred. In this case, a message is created.
- ❹ Establish the second error handler. Any error causes control to be transferred to the label ERROR_HANDLER_2.
- ❺ Prevent accidental execution of the second error handler.

SET ON Command

Use the SET ON command to restore default error handling, or the error handling created by a previous ON command.

Format:

\$ SET ON

SET NOON Command

Use the SET NOON command to disable any error checking performed by the operating system.

- The command interpreter continues to place the status code value in \$STATUS and the severity level in \$SEVERITY, but it does not perform any action based on these values.
- Your command procedure has to detect errors and handle them.

Format:

\$ SET NOON

Guidelines for Using the SET NOON Command

Keep the following points in mind when you use the SET NOON command in a command procedure.

- The SET NOON command is valid at the current command level only. If you use SET NOON in a command procedure that executes another procedure, the default (SET ON) is established while the second procedure executes.
- You cannot use the ON command to handle errors when SET NOON is in effect.
- Default error handling, or the error handling created by a previous ON command, is restored after an EXIT or SET ON command.

When to Use SET NOON

Use SET NOON in a situation where you anticipate that an error might occur, but you do not want your command procedure to stop executing. Some sample situations are:

- Your LOGIN.COM may set your process name. If you are running multiple processes, this results in a fatal error because duplicate process names are not allowed. Code LOGIN.COM so it will keep running even if it fails to set your process name.

```
$ SET NOON
$ SET PROCESS/NAME="may be duplicate"
$ SET ON
```

- You may have a command procedure that deletes your temporary files. If there are no temporary files to delete, you do not want that situation treated as an error.

```
$ SET NOON
$ DELETE TEMP:*.TMP;*
$ SET ON
```

- You want to be able to test for errors in a section of your command procedure and determine how they are to be handled.

```
$ SET NOON
$ FORTRAN 'FILE'      ! Compile a program.
$ IF $SEVERITY THEN LINK 'FILE' ! Link it.
$ IF $SEVERITY THEN RUN 'FILE' ! Run it.
$ SET ON
```

Using \$STATUS and \$SEVERITY in Checking for Errors

If you have disabled error checking with the SET NOON command, you should test the value of \$SEVERITY or \$STATUS after every command that might generate an error.

- An odd (true) value for either of these symbols indicates that the command did not generate an error. To test for an odd value:

```
$ IF $STATUS THEN...
$ IF $SEVERITY THEN...
```

- An even (false) value for either symbol indicates that the command generated an error. To test for an even value:

```
$ IF .NOT. $STATUS THEN...
$ IF .NOT. $SEVERITY THEN...
```

Example: Using \$SEVERITY to Check for Errors

The following example shows how to use \$SEVERITY to check for errors:

```
$! TEST_FORTTRAN.COM
$! Test a VAX FORTRAN program.
$
$ INQUIRE FILE "Filename of source file"
$
$ SET NOON          ! Disable error checking.
$ FORTRAN/LIST 'FILE' ! Compile the source file.
$ IF $SEVERITY      ! Did compile operation succeed?
$ THEN             ! Odd value, compile succeeded.
$ LINK 'FILE'      ! Link the object file.
$ IF $SEVERITY      ! Did link operation succeed?
$ THEN             ! Odd value, link succeeded.
$   ASSIGN/USER SYSS$COMMAND SYSS$INPUT ! Get input for
$   ! next image from the terminal.
$ RUN 'FILE'        ! Try to run the program.
$ IF $SEVERITY      ! Did program run?
$ THEN             ! Odd value, program ran.
$   SET ON          ! Restore error checking.
$   WRITE SYSS$OUTPUT "Program compiled, linked, ran."
$   EXIT            ! Terminate execution.
$ ENDIF
$ ELSE              ! Even value, link failed.
$   SET ON          ! Restore error checking.
$   WRITE SYSS$OUTPUT "Program compiled, but failed to link."
$   EXIT            ! Terminate execution.
$ ENDIF
$ ELSE              ! Even value, compile failed.
$   SET ON          ! Restore error checking.
$   WRITE SYSS$OUTPUT "Compile failed, printing 'FILE'.LIS"
$   PRINT 'FILE'.LIS ! Print the compiler listing.
$   EXIT            ! Terminate execution.
$ ENDIF
$ EXIT
```

Example Notes: Using \$SEVERITY to Check for Errors

This example checks for odd values of \$SEVERITY, indicating that the operation succeeded. In other situations, you may want to check the actual value of \$STATUS or \$SEVERITY if you are interested in specific condition codes.

Handling Ctrl/Y Interrupts

Ctrl/Y Interrupts

By default, if you press Ctrl/Y while a command procedure is executing, the command procedure stops executing and the command interpreter prompts for command input at a special command level called Ctrl/Y command level. If files are open, or some critical operation is in process, you may prefer to perform an orderly exit when a Ctrl/Y interrupt is detected.

Trapping Ctrl/Y Interrupts

Use the ON CONTROL_Y command to control the action of the Ctrl/Y sequence within a command procedure.

Format:

\$ ON CONTROL_Y THEN *command*

- The *command* can be any valid DCL command.

Examples:

```
$ ON CONTROL_Y THEN GOTO CTRL_Y_HANDLER
$ ON CONTROL_Y THEN LOGOUT
```

Action of the ON CONTROL_Y Command

The ON CONTROL_Y command is in effect until one of the following occurs:

- The command procedure exits.
- Another ON CONTROL_Y command is executed.
- The SET NOCONTROL=Y command is executed to disable the Ctrl/Y function.

The action specified by the ON CONTROL_Y command applies **only** within the command procedure in which the command is executed.

- If a lower level command procedure receives a Ctrl/Y interrupt and does not handle it, and there is a Ctrl/Y handler at higher level, the lower level procedure exits back to the higher one and execution resumes at the next command.

The ON CONTROL_Y command is not disabled by a SET NOON command.

Disabling Ctrl/Y Interrupts

Use the SET NOCONTROL=Y command to disable the action of the Ctrl/Y sequence within a command procedure. The **[Interrupt]** message is still displayed, but no interruption takes place. (This command also disables Ctrl/C for all commands and programs that do not have special action routines to respond to Ctrl/C.)

Format:

```
$ SET NOCONTROL=Y
```

Cautions About Disabling Ctrl/Y Interrupts

Keep the following points in mind when deciding whether to use the SET NOCONTROL=Y command in a command procedure.

- The SET NOCONTROL=Y command is for special applications. In general, disabling Ctrl/Y interrupts is not recommended.

For example, if a command procedure that disables Ctrl/Y interrupts begins to loop uncontrollably, you cannot gain control to stop the procedure from your terminal.

- SET NOCONTROL=Y remains in effect after you exit the command level at which you entered the command. You must reset it manually with the SET CONTROL=Y command.

Enabling Ctrl/Y Interrupts

Use the SET CONTROL=Y command to reestablish the default Ctrl/Y action.

Format:

```
$ SET CONTROL=Y
```

Summary

Concepts

Accessing Local and Global Symbol Tables

- An OpenVMS process can have up to 32 command levels.
- Local symbols are available to the command level that created them and all lower command levels. They last for the duration of the command level that defined them, or until they are explicitly deleted.
- Global symbols are available to all command levels. They last for the duration of the process, or until they are explicitly deleted.
- The way a command procedure terminates determines the resulting command level.

Using Symbols in Command Lines

- In some cases DCL automatically substitutes the value of a symbol for the symbol.
- When symbol substitution does not occur automatically, it can be forced by enclosing the symbol in apostrophes.

Writing Command Procedures That Perform Terminal I/O

- DCL provides default values for the logical names SYS\$INPUT, SYS\$OUTPUT, SYS\$ERROR, and SYS\$COMMAND.
- To write information to the terminal, use the WRITE and TYPE commands.
- To get the user to input information to a command procedure, use the INQUIRE and READ commands.

Using Symbols as Variables and Parameters

- Symbols can be used as variables in a command procedure.
- You can supply up to eight parameters, called P1 through P8, when you execute a command procedure. These local symbols represent string data.

Controlling Flow of Execution

- Control the flow of execution of a command procedure by using:
 - IF
 - GOTO
 - @
- The IF-THEN command is used to test a condition and execute statements depending on the outcome of the test.
- The GOTO command can be used to cause a command procedure to branch forward or backward.
- Use the @ command to execute another command procedure from within a command procedure.
- Iteration is the repetition of a group of instructions.
- To write iterative procedures, use a looping structure.

Redirecting Input and Output

- To direct command output to a file, redefine SYS\$OUTPUT.
- To allow the user to edit a file while running a command procedure interactively, redefine SYS\$INPUT.

Using Lexical Functions

- Lexical functions are command language constructs that return and manipulate information about the current process, the system, and arithmetic and string expressions.
- All lexical functions begin with F\$, followed by the function name.
- Arguments for lexical functions are enclosed within parentheses. The parentheses are required even if the lexical function requires no arguments.
- Arguments may be symbols or literal values enclosed in quotation marks.
- Lexical functions can be used to obtain and manipulate character strings.
- Lexical functions can be used to get information about such things as processes, the system, files and devices, logical names, and data types.

Writing Command Procedures That Perform File I/O

- A command procedure can get information from, and write information to, a file.
- To prepare a file for entering or retrieving information, use the OPEN command.
- To enter information into an open file, use the WRITE command.
- To retrieve information from an open file, use the READ command.
- To complete file operations in a way that preserves the integrity of the file, use the CLOSE command.
- A command procedure can anticipate and handle errors that may occur during file I/O.
- When an error occurs, control is transferred to the label specified in the command that generates the error.

Handling Error Conditions and Ctrl/Y

- The command interpreter saves a condition code that tells how each command terminated.
- The system places condition codes in the global symbol \$STATUS.
- The ON command allows you to determine the action the system takes when an error condition occurs.
- The SET NOON command tells the system not to respond to error conditions. The command procedure must detect and handle all errors. The SET ON command restores the default error handling.
- A command procedure can trap or disable Ctrl/Y interrupts.
- Use the ON CONTROL_Y command to control the action of the Ctrl/Y sequence within a command procedure.
- Use the SET NOCONTROL=Y command to disable the action of the Ctrl/Y sequence within a command procedure.

Commands

Accessing Local and Global Symbol Tables

= or :=

Creates a local symbol.

== or :=

Creates a global symbol.

EXIT

Terminates the command procedure and returns to the previous command level.

STOP

Terminates the command procedure and returns to DCL command level (command level 0).

Using Symbols in Command Lines

'symbol-name'

Forces symbol substitution.

" 'symbol-name' "

Forces symbol substitution inside a character string.

Writing Command Procedures That Perform Terminal I/O

TYPE

Displays information at a terminal.

WRITE

Displays information at a terminal.

INQUIRE

Gets information from a user.

READ

Gets information from a user.

Using Symbols as Variables and Parameters

@filename.com parameter1 ... parameter8

Passes as many as eight parameters to a command procedure.

Controlling Flow of Execution

IF-THEN-ELSE

Executes one set of statements if the conditional expression evaluates true, and another set of statements if the condition is false.

GOTO

Causes execution to branch to the specified label.

@

Starts another command level from within a command procedure or from the DCL prompt.

Redirecting Input and Output

ASSIGN

Creates a logical name and associates an equivalence string with the logical name.

DEASSIGN

Cancels a logical name assignment.

DEFINE

Creates a logical name and associates an equivalence string with the logical name.

ASSIGN

Creates a logical name and associates an equivalence string with the logical name.

DEASSIGN

Cancels a logical name assignment.

DEFINE

Creates a logical name and associates an equivalence string with the logical name.

Writing Command Procedures That Perform File I/O

OPEN

Opens a file for input or output.

WRITE

Writes a record to a file.

READ

Gets a record from a file.

CLOSE

Closes a file opened for input or output.

OPEN/ERROR=label

READ/ERROR=label

WRITE/ERROR=label

CLOSE/ERROR=label

Transfers control to the specified label if an error occurs in a file I/O operation.

READ/END_OF_FILE=label

Transfers control to the specified label if an error occurs in a file read operation.

Handling Error Conditions and Ctrl/Y

ON

Specifies an action to be performed when a condition occurs.

SET NOON

Disables error checking.

SET ON

Reestablishes error checking.

SET CONTROL=Y

Enables Ctrl/Y interrupts.

SET NOCONTROL=Y

Disables Ctrl/Y interrupts.

Managing System Users

Module 3

About This Module

Introduction

A major system management function is the management of users. A system manager is responsible for such tasks as:

- Controlling access to the system
- Maintaining system security
- Defining user access times
- Controlling user processes

This module introduces the basic tasks involved in managing the users of an OpenVMS system.

Objectives

To manage users on an OpenVMS system, a system manager should be able to:

- Use the Authorize utility.
- Use the SHOW and LIST commands of the Authorize utility to list user accounts on an OpenVMS system.
- Add a user to the system.
- Remove a user from the system.
- Manage and monitor user processes.
- Manage the use of disk space.

Overview

The Authorize utility is used to manipulate the User Authorization File (UAF), which defines the system users.

Topics

In this section the following topics are discussed:

- User Authorization File (UAF)
- Running the Authorize utility
- Getting help from the Authorize utility
- Displaying a UAF record

User Authorization File

The system manager defines user names in the user authorization file (UAF).

- The UAF contains one record for each user authorized to log in to the system.
- Users are identified on each OpenVMS system by their user names.

Each UAF record consists of the user name, an encoded password, and several other fields.

- Each time you log in, the system reads the record corresponding to your user name from the UAF.
- The system creates a process for you (if you entered the correct password) with the limits and restrictions found in the various fields of your UAF record.
- The system manager can modify any field in a UAF record.



Note

If you inherit a system for which you do not know any passwords you can reboot it with UAFALTERNATE set to 1 or reboot it with the startup file pointing to OPA0:

Running the Authorize Utility

Use the Authorize utility to manipulate the User Authorization File (UAF). You must be logged in to the SYSTEM account or an account with SYSPRV to run this utility.

Running the Authorize Utility

To run the Authorize utility:

- Give yourself the SYSPRV privilege.
- Set default to the SYS\$SYSTEM directory.
- Type RUN AUTHORIZE to start the utility.
- Type EXIT or press Ctrl/Z to exit from the utility.

Example: Running the Authorize Utility

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF>
.
.
.
UAF> EXIT
$
```



Note

Use of the Authorize utility requires write access to any or all of the following files in SYS\$SYSTEM:

- SYSUAF.DAT
- NETPROXY.DAT

On OpenVMS VAX systems, the default network proxy authorization file is NET\$PROXY.DAT. However, the Authorize utility maintains the file

- NETPROXY.DAT
 - RIGHTSLIST.DAT:
-

Getting Help from the Authorize Utility

Use the HELP command in the Authorize utility to obtain information about AUTHORIZE commands and qualifiers.

Example: Getting Help from the Authorize Utility

The following example shows how to use the HELP command:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF> HELP
```

Information available:

ADD	Command_Summary	COPY	CREATE	DEFAULT	EXIT
GRANT	HELP	LIST	MODIFY	REMOVE	RENAME
SHOW	Usage_Summary				REVOKE

Topic?

.
.
.

UAF>

Displaying a UAF Record

Use the SHOW command in the Authorize utility to display individual UAF records.

Displaying a UAF Record

To display individual UAF records:

1. Set default to the SYS\$SYSTEM directory.
2. Type RUN AUTHORIZE to start the utility.
3. Use the SHOW command to display a record.

Example: Displaying a UAF Record

The following example shows how to display a user record:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF> SHOW username
UAF>
```



Note

- Some of the UAF fields contain information unique for each user, such as the OWNER, UIC, and DEFAULT fields.
- Other fields contain information common to users in the same group, such as the ACCOUNT and LGICMD fields.

Only the fields discussed in this section have been called out in the notes for the example.

Example: Displaying a UAF Record

The following example uses the AUTHORIZE SHOW command to display the UAF record for a user named USER1.

```
$ SET DEFAULT SYS$SYSTEM
```

```
$ RUN AUTHORIZE
```

```
UAF> SHOW USER1
```

```

❶ Username: USER1                ❷ Owner: Admin User
❸ Account:  GRP11                ❹ UIC:  [11,2] ([GRP11,USER1])
  CLI:      DCL                    Tables: DCLTABLES
❺ Default:  DISK$USER:[USER1]
❻ LGICMD:   SYS$MANAGER:GRP11LOGIN
❼ Flags:    Diswelcome Disnewmail
  Primary days: Mon Tue Wed Thu Fri
  Secondary days:                               Sat Sun
  Primary 000000000001111111112222    Secondary 000000000001111111112222
  Day Hours 012345678901234567890123    Day Hours 012345678901234567890123
  Network:  ----- No access -----          ----- Full access -----
  Batch:    #####-----#####              -----#####-----
  Local:    ##### ----- #####              ##### ----- #####
  Dialup:   ----- Full access-----          ----- No access -----
  Remote:   ----- Full access-----          ----- No access -----
  Expiration: (none)      Pwdminimum: 8 Login Fails: 0
  Pwdlifetime: 90 00:00    Pwdchange: 16-OCT-1999 09:08
  Last Login: 7-NOV-1999 12:06 (interactive), (none) (non-interactive)
❽ Maxjobs: 0 Fillm: 300 Bytln 75000
  Maxacct jobs: 0 Shrfillm: 0 Pbytln: 0
  Maxdetach: 0 BIOln: 10 JTquota: 1024
  Prclm: 10 DIOln: 200 WSdef: 256
  Prio: 4 ASTln: 200 WSquo: 512
  Queprio: 0 TQElm: 200 WSextent: 2048
  CPU: (none) Enqlm: 200 Pgflquo: 40960
❾ Authorized Privileges:
  TMPMBX NETMBX
  Default Privileges:
  TMPMBX NETMBX
❿ Identifier                ⓫ Value    ⓬ Attributes
  PROJECT X                  %X8001001E  RESOURCE NODYNAMIC
  DOCU_PROC                  %X80010044  NORESOURCE NODYNAMIC
UAF>
```

Example Notes: Displaying a UAF Record

Notes on the Displaying a UAF Record example include:

- ❶ This is the user name that is typed in response to the **Username:** prompt when logging in.
- ❷ The owner field is not used by the system. To include spaces in the name, enclose it in quotation marks ("").
- ❸ Used by the Accounting utility to identify processes whose activities are billed in common. To include spaces in the account name, enclose it in quotation marks ("").
- ❹ The User Identification Code (UIC) is used to enforce protection of devices, volumes, and files. It also regulates the ability of the process to communicate with and affect other processes. The UIC may be displayed in a numerical or character string form. The numeric form is [group,member]. The character string form is [account,username].
- ❺ Used by the operating system to establish the default device and directory at login time. It is preferable to identify the default device by a logical name based on the volume label (such as DISK\$USER), rather than by a physical device name (such as DUA1).
- ❻ Specifies a systemwide or groupwide login command file. Commands in this file are executed before the user can issue commands interactively or through a batch command file. If this field is left blank, the system executes the LOGIN.COM file in the user's default directory before executing user commands.
- ❼ Special restrictions during and after logging in to the system. These flags also include DISUSER, used for disabling an account.
- ❽ Limits used to restrict the use of various resources by a process. Limits are also called quotas, not to be confused with disk quotas.
- ❾ Privileges assigned to the user. Default privileges are given to the user at login. Authorized privileges are given when the user enters the SET PROCESS/PRIVILEGE command.
- ❿ ACL identifiers that the user holds and that are recorded in the rights database file.
- ⓫ Characteristics specified when adding identifiers to the rights database or when granting identifiers to users.

Using the Authorize Utility to List User Accounts

Overview

The system manager may need to list the user accounts on the system. Reasons for this include:

- Before adding a new user, it is important to select a UIC and a user name that are not already assigned to another user.
- Listing existing accounts enables the system manager to assign a new user an appropriate UIC group number.
- For quick reference, it is helpful to keep a current listing of the users on the system.

Topics

In this section the following topics are discussed:

- Using the Authorize utility to list user accounts

Using the Authorize Utility to List User Accounts

Listing User Accounts

Use the Authorize utility to list user records, either at your terminal or in a file.

- The LIST command creates a file called SYSUAF.LIS, which contains a list of user records in the UAF.
- The SHOW/BRIEF command displays a list of user records at the terminal.

Sample Commands for Listing User Accounts

The following table lists commonly used AUTHORIZE LIST and SHOW commands.

If you want a list of UAF records for...	Use this command to write the list to a file:	Use this command to display the list at your terminal:
All users	LIST	SHOW/BRIEF *
A specific UIC group	LIST <i>[group,*]</i>	SHOW/BRIEF <i>[group,*]</i>
A specific user	LIST <i>username</i>	SHOW/BRIEF <i>username</i>



Note

Passwords are never listed.

Information Contained in a Brief UAF Listing

A brief listing of UAF records contains the following information:

- Account owner
- User name
- UIC
- Account name
- Privileges
- Process priority
- Default disk and directory

Example: Brief UAF Listing

The following example shows the use of the SHOW/BRIEF command and the LIST command.

```
$ RUN AUTHORIZE
```

```
①UAF> SHOW/BRIEF [400,*]
```

Owner	Username	UIC	Account	Privs	Pri	Directory
Group leader	PER_USER1	[400,1]	PER	All	4	
\$1\$DUA1:[PER_USER1]						
Jerry D.	PER_USER2	[400,2]	PER	Normal	4	
\$1\$DUA1:[PER_USER2]						
Krystle R.	PER_USER4	[400,4]	PER	Normal	4	
\$1\$DUA1:[PER_USER4]						
Dave L.	PER_USER5	[400,5]	PER	Normal	4	
\$1\$DUA1:[PER_USER5]						
Min R.	PER_USER6	[400,6]	PER	Normal	4	
\$1\$DUA1:[PER_USER6]						
.						
.						
.						

```
②UAF> LIST [400,*]
```

```
UAF-I-LSTMSG1, writing listing file
%UAF-I-LSTMSG2, listing file SYSUAF.LIS complete
UAF>
```

Example Notes: Brief UAF Listing

Notes on the Brief UAF Listing example include:

- ① Use the Authorize utility to display a brief listing of all accounts having the UIC group number 400.
- ② Create a file containing the same information.



Note

Privileges fall into seven categories according to the damage the user possessing them can cause the system:

Category	Damage User Could Cause System
None	No privileges
Normal	Minimum privileges to use the system effectively
Group	Potential to interfere with members of the same group
Devour	Potential to consume noncritical systemwide resources
System	Potential to interfere with normal system operation
Objects	Potential to compromise object security
All	Potential to control the system

Adding a User to the System

Overview

The procedure for adding a system user can vary widely from site to site, depending on the site's security policies. When new user accounts are added, you need to consider how the user will use the system and what privileges the user will need.

Topics

In this section the following topics are discussed:

- Planning the User's Account
- Establishing a Disk Quota
- Creating a Default Directory
- Testing the New Account
- Using an Automated Procedure to Add a User
- Using OpenVMS Management Tools for Windows NT to add User Accounts
- Common Errors in Adding a User

Planning the User's Account

Planning the Account

Before adding a user to the system, decide on:

- User name
- Password
- User identification code (UIC)
- Default device
- Default directory
- Account name

Individual site requirements may include other parameters that should be established when a user is added.

Assigning a User Name

The user name identifies the user to the operating system at login time.

- The user name usually matches the last name of the user, or some combination of the user's first and last names. If security is a serious problem, more cryptic user names may need to be assigned.
- The user name is a string of 1 to 12 characters, including letters, numbers, and underscores (_). Dollar signs (\$) are permitted, but they are usually reserved for system names. Fully numeric user names should not be assigned.



Note

While fully numeric user names are permitted, fully numeric identifiers are not. Numeric user names should be avoided because they do not receive corresponding identifiers.

Assigning a Password

The password serves as an authenticator that should be known only to the user and the operating system.

- By default, passwords assigned by the system manager using the Authorize utility are pre-expired, and a new password must be chosen by the user the first time the user logs in.
- Passwords can contain from 0 to 31 characters and can include letters, numbers, dollar signs (\$), and underscores (_).
- Passwords generated by the system contain 1 to 10 characters.
- Short or obvious passwords present a potential security problem by making it easy for an unauthorized person to log in to the system. The system manager should discuss this concern with the new user.
- The default minimum length for passwords set by the user is eight characters. Some sites require a longer password.



Note

The system manager can use the Authorize utility to assign a password that is shorter than the required minimum length. Password length is enforced only for the DCL command SET PASSWORD.

Minimum Password Length

Use the Authorize utility to specify a minimum password length with the PWDMINIMUM qualifier when you add or modify the user's record.

Example:

```
UAF> MODIFY USER1/PWDMINIMUM=12
```

Assigning a User Identification Code

The user identification code (UIC) indicates the user's membership in a group that corresponds to the user's department, project, or function.

- This code is a group and member number specified in octal, separated by a comma, and enclosed in brackets. Example: [11,2]
 - The group number must be in the range 1 to 37776 (octal). Group numbers 10 (octal) and lower should be reserved for system accounts.
 - The member number must be in the range 0 to 177776 (octal).
- The UIC should be unique on the system because it is used to determine ownership of the user's files.

Assigning a Default Device

Select a device that has sufficient space for the number of files the user is likely to have. It is helpful (but not necessary) for all members of the same project group to use the same disk as their default device.

Assigning a Default Directory

The user's top-level (default) directory will be located on the default device.

- It is customary, but not required, for the name of the default directory to match the user name.
- The default directory cannot have the same name as an existing top-level directory on that device.

Assigning an Account Name

The account name is a 1- to 8-character string.

- This name can be used as a billing name or number by the Accounting utility.
- The account is used as the first (group) part of the user's alphanumeric UIC, and should match that of other members of the user's UIC group.

Adding the UAF Record

Use the AUTHORIZE command ADD to add a record to the UAF.

- The UIC should be unique on the system. The Authorize utility returns an error if a nonunique value is entered.
- Parameters not specified take their default values.
- If the DEFAULT record has the DISUSER flag set, use the NODISUSER flag to specify that this user's account is not disabled.



Note

The Authorize utility returns an error if a nonunique UIC value is entered.

- The user record is still added to the UAF.
 - The identifier is not added to the rights database.
-

Specifying the Account When Adding a User

If this is the first user to be assigned to this UIC group, specify the account on the ADD command line so the identifier will be created properly.

Enabling New User Account

The user record will inherit the flag DISUSER from the DEFAULT account if this flag has not been cleared from the DEFAULT UAF record.

Default UAF Record

If no value is specified for a field in a new UAF record, the value for that field is taken from the DEFAULT record in the User Authorization File.

Example: Manually Adding the User Record

The following example shows the use of the Authorize utility to add a record for USER1.

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
❶ UAF> ADD USER1/UIC=[11,2]/ACCOUNT=GRP11-
❷ UAF> /DEVICE=DISK$USER/DIRECTORY=[USER1]-
  _UAF> /PASSWORD=ENIGMA/OWNER="Admin User"-
  _UAF> /FLAG=NODISUSER
❸ %UAF-I-ADDMSG, user record successfully added
❹ %UAF-I-RDBADDMSGU, identifier USER1 value: [000011,000002] added to
rights data base
  UAF> EXIT
❺ %UAF-I-DONMSG, system authorization file modified
  %UAF-I-NAFNOMODS, no modifications made to network proxy data base
  %UAF-I-RDBDONMSG, rights data base modified
$
```

Example Notes: Manually Adding the User Record

Notes on the Manually Adding the User Record example include:

- ❶ Note the use of the hyphen character (-) to continue the command.
- ❷ The Authorize utility gives the _UAF> prompt to indicate that the command has not been terminated.
- ❸ Informational message from the Authorize utility.
- ❹ The user has been given a unique identifier in the system rights database.
- ❺ When you exit from the Authorize utility, these messages indicate what (if any) system files have been changed.

Establishing a Disk Quota

Disk Quota

A system manager can use disk quotas to limit the amount of space a user can use on a particular disk volume. If quotas are enabled on a disk volume that the user will use, a disk quota must be established for that user's UIC on that volume.

Each UIC is assigned a permanent quota and an overdraft value.

- The maximum number of blocks the UIC is permitted to consume on the specified disk is the sum of the permanent quota and overdraft value.
- For the user to be able to create a new file, the user's usage count must be less than the permanent quota.
- The user may extend an existing file until the usage exceeds the sum of the permanent and overdraft quotas.



Note

Only users with EXQUOTA privilege can bypass disk quota restrictions.

Using SYSMAN to Set Up a Disk Quota

The System Management utility (SYSMAN) centralizes system management so nodes or clusters can be managed from one location.

Run the SYSMAN utility to add the user's UIC to the quota file on each volume where quotas are enabled and where the user will need to create or modify files.

SYSMAN requires the following privileges:

- OPER on the local node
- Authorization for OPER or SETPRV on any remote node being accessed
- Privileges required by individual commands being executed

Example: Using SYSMAN to Set Up a Disk Quota

In the following example, the UIC of [11,2] is assigned a permanent quota of 500 blocks and an overdraft of 100 blocks on the volume DISK\$USER.

```
$ SET PROCESS/PRIVILEGE=OPER
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA ADD [11,2]/DEVICE=DISK$USER-
_SYSMAN> /PERMQUOTA=500/OVERDRAFT=100
SYSMAN> EXIT
$
```

Example: Default Disk Quota Values

Default values for permanent quota and overdraft are taken from the UIC entry [0,0].

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA SHOW [0,0]/DEVICE=WORK12:

%SYSMAN-I-QUOTA, disk quota statistics on device WORK12: -- Node TIDY
      UIC      Usage      Permanent Quota      Overdraft Limit
[0,0]          0          1000             100
SYSMAN>
```



Note

SYSMAN can do many things besides setting up disk quotas, and this is not intended to be an exhaustive coverage of the utility.

Overbooking a Disk

Disk space can be overbooked, meaning that the sum of all disk quotas exceeds the total number of blocks on the disk. To determine how much to overbook a disk, monitor the space in use as a percentage of the space booked.

Disk space should not be overbooked when both of the following are true:

- Users do not have enough disk space and always use close to their quota.
- Disks are run at more than ninety percent of their capacity.

Creating a Default Directory

Default Directory

When a user logs in, the operating system looks at the UAF record to determine what device and directory should be used as a default for that user.



Note

You can create the disk directory before or after the UAF record is created.

Setting Up a Default Directory

When adding a user to the system, create a directory that matches the DEFAULT field in the UAF record.

- Create the user's default directory on a volume where the user will be able to create and modify files. (This could even be a volume where quotas are disabled.)
- This should be the same directory specified in the UAF using the /DEVICE and /DIRECTORY qualifiers.
- Use the DCL CREATE/DIRECTORY command, specifying the UIC of the owner of the directory.

Example of specifying the numeric UIC of the owner

```
$ CREATE/DIRECTORY DISK$USER:[USER1]/OWNER_UIC=[11,2]  
$
```

Example of specifying the identifier of the owner:

```
$ CREATE/DIRECTORY DISK$USER:[USER1]/OWNER_UIC=[USER1]  
$
```



Note

Students may be confused by the identifier, [USER1], because it looks like a directory name. Explain that the identifier matches the user name, and a user's default directory often also matches the user name.

Testing the New Account

Once a user has been added to the system, the system manager tests the account to ensure that all flags and fields have been set correctly in the UAF, and that the proper disk quotas have been established.

Example: Testing a New Account

The following example shows a system manager testing the new account created for USER1.

```
❶ Username: USER1
❷ Password
    Welcome to the Production System

❸ Your password has expired; you must set a new password to log in

❹ New password:
    Verification:

❺ $ SHOW PROCESS

    23-NOV-1999 11:44:12.96  User: USER1      Process ID:    2020071
                           Node: TIDY       Process name:  "USER1"

    Terminal:             TTC1: (TIDY::SYSTEM)
    User Identifier:      [USER1]
    Base priority:        4
    Default file spec:    DISK$USER:[USER1]

    Devices allocated:    TIDY$TTC1:

❻ $ SHOW DEFAULT
    DISK$USER:[USER1]

❼ $ SHOW QUOTA
    User [GRP11,USER1] has 6 blocks used, 494 available,
    of 500 authorized and permitted overdraft of 100 blocks on DISK$USER
$ LOGOUT
    USER1      logged out at 23-NOV-1999 11:53:24.50
```

Example Notes: Testing a New Account

Notes on the Testing a New Account example include:

- ❶ Log in to the account using the user name that appears in the UAF.
- ❷ The password will not appear in the UAF.
- ❸ By default, the account has a pre-expired password. This forces the user to select a new password at the first login.
- ❹ The new password is subject to system checks for minimum length, password history, and presence in the system dictionary.
- ❺ Use SHOW PROCESS to check the UIC.
- ❻ Use SHOW DEFAULT to display the default login directory.
- ❼ Use SHOW QUOTA to check the disk quota file entry.

Resetting the User's Password

When logging in to test an account that has a pre-expired password, the system manager must set a new password for the account.

This new password is not pre-expired. It should be reset using the Authorize utility so the user will be forced to change it when he or she logs in for the first time.

Passwords set using the Authorize utility are not subject to the same checks as those established with the SET PASSWORD command.



Note

The system manager is not required to reset the password, but resetting it with the Authorize utility forces the user to change the password.

Example: Resetting the User's Password

The following example shows the system manager logging in to a new account and resetting the user's password.

```
❶ $ SET HOST 0

❷ Username: USER1
   Password:
      Welcome to the Production System

❸ Your password has expired; you must set a new password to log in

❹ New password:
   Verification:

❺ $ LOGOUT
   USER1      logged out at 23-NOV-1999 12:38:56.81
   %REM-S-END, control returned to node _NEAT::
   $ SET DEFAULT SYS$SYSTEM

❻ $ RUN AUTHORIZE
❼ UAF> MODIFY USER1/PASSWORD=ENIGMA
   %UAF-I-MDFYMSG, user record(s) updated
   UAF>
```

Example Notes: Resetting the User's Password

Notes on the Resetting the User's Password example include:

- ❶ Use the SET HOST command to log in to the system in order to test the user's account. If SET HOST is not available on your system, log out and log back in to test the account.
- ❷ Enter the new user's user name and password.
- ❸ If the account has a pre-expired password, a new password must be chosen the first time someone logs in to the account.
- ❹ The new password is subject to the system's restrictions on password length, history, and presence in the system's password dictionary.
- ❺ Log out of the new account and return control to the system manager's process.
- ❻ Run the Authorize utility to reset the user's password.
- ❼ The new password can be the same as the one originally assigned. Passwords set using the Authorize utility are not subject to the system's restrictions.

Using an Automated Procedure to Add a User

ADDUSER.COM

The ADDUSER.COM command procedure, which is supplied with the OpenVMS operating system, helps the system manager add users to the system. The command procedure prompts for the necessary fields and flags and manipulates the UAF to add an account.

Example: ADDUSER Session

The following example shows the use of ADDUSER.COM to add a user to the system.

```

❶ $ @SYS$EXAMPLES:ADDUSER

*****
*   Creating a NEW user account... If at ANY TIME you need help about a   *
*   prompt, just type "?". *
*****

❷ Username(s) - separate by commas: USER1

*** Processing USER1's account ***

❸ Full name for USER1: Admin User
❹ Password (password is not echoed to terminal) [USER1]:

❺ UIC Group number [200]: 11

❻ UIC Member number: 2

❼ Account name: ?

You must enter the account name to be used for the user. This is generally
used for billing purposes.

❽ Account name: GRP11
❾ Privileges [TMPMBX,NETMBX]: TMPMBX,NETMBX,OPER

❿ Login directory [USER1]:

⓫ Login device [SYS$SYSDEVICE]: DISK$USER:

%CREATE-I-CREATED, DISK$USER:[USER1] created
%UAF-I-ADDMSG, user record successfully added
%UAF-I-RDBADDMSGU, identifier USER1 value: [000011,000002] added to rights
database
%UAF-I-RDBADDMSGU, identifier GRP11 value: [000011,177777] added to rights
database
%UAF-I-DONEMSG, system authorization file modified
%UAF-I-NAFNOMODS, no modifications made to network proxy database
%UAF-I-RDBDONEMSG, rights database modified

```

(Continued)

ADDUSER Session (continued)

Check newly created account:

```
Username: USER1           Owner:   Admin User
Account:  GRP11           UIC:    [11,2] ([GRP11,USER1])
CLI:      DCL             Tables:  DCLTABLES
Default:  DISK$USER:[USER1]
LGICMD:   LOGIN
Flags:
Primary days:  Mon Tue Wed Thu Fri
Secondary days:                Sat Sun
No access restrictions
Expiration:          (none)      Pwdminimum: 15      Login Fails:      0
Pwdlifetime:         180 00:00   Pwdchange:      (pre-expired)
Last Login:          (none) (interactive),          (none) (non-
interactive)
Maxjobs:             0 Fillm:      100 Byt1m:         36000
Maxacctjobs:         0 Shrfillm:    0 Pbyt1m:         0
Maxdetach:           0 BIO1m:      40 JTquota:        1024
Prclm:               10 DIO1m:     100 WSdef:         512
Prio:                4 AST1m:      600 WSquo:        2048
Queprio:             0 TQE1m:     10 WSextent:        6124
CPU:                 (none) Enqlm:  100 Pgflquo:       30000
Authorized Privileges:
  TMPMBX OPER NETMBX
Default Privileges:
  TMPMBX NETMBX
%UAF-I-NOMODS, no modifications made to system authorization file
%UAF-I-NAFNOMODS, no modifications made to network proxy database
%UAF-I-RDBNOMODS, no modifications made to rights database
```

```
12 Is everything satisfactory with the account [YES]:
$
```

Example Notes: ADDUSER Session

Notes on the ADDUSER Session example include:

- ❶ Execute the ADDUSER command procedure.
- ❷ Enter one or more user names. This will be the name under which the user will log in.
- ❸ Not used by the system, but included for the convenience of the system manager.
- ❹ You will not see the password as you type it in.
- ❺ The group number must be an octal number.
- ❻ The member number must be in octal and must be unique within the selected group.
- ❼ Here a question mark is entered to see help information on account names.
- ❽ This name is used by accounting packages to track resource usage.
- ❾ This account will carry OPER privileges as well as TMPMBX and NETMBX. These are authorized, not default, privileges. To activate OPER, USER1 must use the SET PROCESS/PRIVILEGE command.
- ❿ Also known as the home directory.
- ⓫ Use a logical name instead of a physical device name for the login device. This will save you time later if you need to transfer users to a different physical device.
- ⓫ If something in the account is incorrect or missing, answer NO to step through the process again.

Using OpenVMS Management Tools for Windows NT to Add a User Account

What Is OpenVMS Management Station?

OpenVMS Management Station is a powerful Microsoft Windows and Windows NT based management tool for system managers and others who perform management tasks on OpenVMS systems. OpenVMS Management Station provides a comprehensive user interface to OpenVMS account, printer, and storage management.

OpenVMS Management Station allows you to organize the systems you need to manage in ways that are meaningful to you and your environment, and allows you to manage user accounts, printers, and storage on those systems.

Plus, you can easily manage user accounts, printers, and storage across multiple OpenVMS systems. For example, assume that you have an account on three different OpenVMS Cluster systems. With OpenVMS Management Station, you can easily update a process quota, add a privilege, grant a rights identifier for each instance of the account.

OpenVMS Management Station consists of two components. You first install the client software on a PC to perform all management operations. You then install the server component on all of the OpenVMS systems you want to manage. You do not interact directly with the server component.

Account Management

You can use OpenVMS Management Station to manage OpenVMS user accounts in a convenient, easy manner. For example, when creating an account, OpenVMS Management Station can add a user authorization file (UAF) entry, grant rights identifiers, create an OpenVMS directory, set a disk quota, and set up OpenVMS Mail characteristics.

Plus, you can easily **manage user accounts across multiple OpenVMS systems**. For example, assume that you have an account on three different OpenVMS Cluster systems. OpenVMS Management Station makes it easy to update a process quota, add a privilege, grant a rights identifier, and so forth, for each instance of your account.

OpenVMS Management Station can manage the following OpenVMS resources:

- SYSUAF.DAT user authorization file
- TNT\$UADB.DAT file
- RIGHTSLIST.DAT user rights file
- Network proxy database
- Account login directory trees
- User account disk quotas
- OpenVMS Mail VMSMAIL_PROFILE.DATA file

Common Errors in Adding a User

Common Errors

There are several common errors that are easy to make when adding a user to the system. These include:

- Not having sufficient privilege to run AUTHORIZE
- Not running AUTHORIZE correctly
- Assigning a duplicate UIC
- Not specifying account for a new UIC group
- Not enabling the user's account
- Not specifying a default device/directory
- Not specifying owner of directory
- Incorrectly locating the user's directory
- Not setting up a disk quota

Error: Not Having Sufficient Privilege to Run AUTHORIZE

To access the system UAF, the system manager must have either SYSPRV privilege or a system UIC.

Example of error:

```
$ SET DEF SYS$SYSTEM
$ RUN AUTHORIZE
%UAF-E-NAOFIL, unable to open system authorization file (SYSUAF.DAT)
-RMS-E-PRV, insufficient privilege or file protection violation
$
```

Correcting: Not Having Sufficient Privilege to Run AUTHORIZE

To correct this error:

```
$ SET PROCESS/PRIVILEGE=SYSPRV
$ RUN AUTHORIZE
.
.
.
```



Note

If the system manager has SETPRV privilege, the SYSPRV privilege can be enabled.

Error: Not Running AUTHORIZE Correctly

The Authorize utility looks in the default directory for SYSUAF.DAT, the User Authorization File.

- If the system manager does not set default to SYS\$SYSTEM before running AUTHORIZE, the system UAF will not be found.
- AUTHORIZE offers the option of creating a UAF in the default directory if it is not found. Although this file is useful for practice, the system manager may accidentally create a real user's record in this file.

Example of creating a practice UAF:

```
$ RUN SYS$SYSTEM:AUTHORIZE
%UAF-E-NAOFIL, unable to open system authorization file (SYSUAF.DAT)
-RMS-E-FNF, file not found
Do you want to create a new file? Y
UAF>
```

If the user's record is added to this practice file, the user name will not be found in the system UAF when the user attempts to log in.

Personnel System - Unauthorized Access Prohibited

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

Correcting: Not Running AUTHORIZE Correctly

To correct this error:

1. Set default to SYS\$SYSTEM
2. Run AUTHORIZE again
3. Add the user's record



Note

There may be a system logical name, SYSUAF, that points to the system UAF file.

- If this logical name has been assigned, the user authorization file will be found even if the default has not been set to SYS\$SYSTEM.
 - To create a practice UAF file, the system manager must create a process logical name to override this system logical.
-

Error: Assigning a Duplicate UIC

A duplicate UIC can be assigned in several ways.

- Assigning a UIC without checking a listing of existing UICs
- Adding a user by copying another user's UAF record, and not specifying a different UIC
- Adding a user without specifying a UIC, so the UIC of the DEFAULT record is assigned

Example of adding a user without specifying a UIC:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF> ADD PER_USER7/PASSWORD=TESTING
%UAF-I-ADDMSG, user record successfully added
%UAF-E-RDBADDERRU, unable to add PER_USER7 value: [000200,000200] to rights data
base
-SYSTEM-F-DUPIDENT, duplicate identifier
UAF> SHOW PER_USER7

Username: PER_USER7                      Owner:
Account:                                UIC:  [200,200] ([SYS])
CLI:      DCL                           Tables:  DCLTABLES
Default:  $1$DUA2:[USER]
.
.
.
UAF> EXIT
%UAF-I-DONEMSG, system authorization file modified
%UAF-I-RDBNOMODS, no modifications made to rights data base
$
```

Correcting: Assigning a Duplicate UIC

To correct this error:

1. Run Authorize
2. Modify UIC field

Example of correcting the error:

```
UAF> ADD/IDENTIFIER/VALUE=UIC:[400,7] PER_USER7
%UAF-I-RDBADDMSGU, identifier PER_USER7 value: [000400,000007] added to
rights data base
UAF> MODIFY PER_USER7/UIC=[400,7]
%UAF-E-RDBMDFYERR, unable to modify identifier DEFAULT
-SYSTEM-F-DUPIDENT, duplicate identifier
%UAF-I-MDFYMSG, user record(s) updated
UAF> SHOW PER_USER7

Username: PER_USER7                      Owner:
Account:                                UIC:  [400,7] ([PER_USER7])
.
.
.
UAF>
```

After the user UIC is set up correctly, use the MODIFY command to establish other UAF fields.

Error: Not Specifying Account for a New UIC Group

When adding a user with a UIC group that has not previously been assigned, it is important to specify the account for that group so the identifier will be set up correctly.

Example of not specifying account:

```
UAF> ADD FIN_USER1/UIC=[500,1]/PASSWORD=TESTING
%UAF-I-ADDMSG, user record successfully added
%UAF-I-RDBADDMSGU, identifier FIN_USER1 value: [000500,000001] added to rights data
base
UAF> SHOW FIN_USER1

Username: FIN_USER1          Owner:
Account:                    UIC:  [500,1] ([FIN_USER1])
.
.
.
UAF>
```

In the example, the named UIC has only one part.

Correcting: Not Specifying Account for a New UIC Group

To correct this error, run AUTHORIZE:

- Modify the user's UIC group

After the user identifiers are set up correctly, use the MODIFY command to establish other UAF fields such as device and directory.



Note

At a minimum, you should specify account and UIC on the ADD command line when adding a user to the system.

Although there are other ways of adding the identifier without removing the user, it is easier to remove and start over.

Example of specifically adding an identifier:

```
UAF> ADD/IDENTIFIER/VALUE=UIC:[400,*] PER
%UAF-I-RDBADDMSGU, identifier PER value: [000400,177777]
added to RIGHTSLIST.DAT
UAF>
```

Error: Not Enabling the User's Account

If the DISUSER flag is set for the DEFAULT UAF record, a new user record will inherit this flag and the new account will be disabled.

Example of trying to log in to a disabled account:

```
Personnel System - Unauthorized Access Prohibited

Username: USER1
Password:
User authorization failure
```

Correcting: Not Enabling the User's Account

To correct this error:

1. Run AUTHORIZE
2. Remove the DISUSER flag from the user's UAF record

```
UAF> MODIFY USER1/FLAG=NODISUSER
%UAF-I-MDFYMSG, user record(s) updated
UAF>
```

Error: Not Specifying a Default Device/Directory

If values for the user's default device or directory are not specified in the UAF record, they will default to the values in the DEFAULT record.

Example of the error:

```
$ DIRECTORY
%DIRECT-E-OPENIN, error opening $1$DUA0:[USER]*.*;* as input
-RMS-E-DNF, directory not found
-SYSTEM-W-NOSUCHFILE, no such file
$
```

Correcting: Not Specifying a Default Device/Directory

To correct this error:

1. Run AUTHORIZE and
2. Modify the user's record, specifying the correct device (Logical names can be used for devices with AUTHORIZE.)

```
UAF> MODIFY USER1/DEVICE=WORK12/DIRECTORY=[USER1]
%UAF-I-MDFYMSG, user record(s) updated
UAF>
```



Note

Remind students that WORK12 is a logical name that points to \$1\$DUA1:. Use logical names in the UAF record rather than hard coding a device name. If the disk has to be changed, the system manager only needs to change the logical name assignment.

Error: Not Specifying Owner of Directory

If the user's UIC is not specified as the owner of the user's default directory, the user will not be able to write to the directory due to protection violations.

Example of the system manager's error:

```
$ CREATE/DIRECTORY WORK12:[USER7]
$
```

Example of the user's problem:

```
$ CREATE TEST.DAT
%CREATE-E-OPENOUT, error opening WORK12:[USER7]TEST.DAT; as output
-RMS-E-PRV, insufficient privilege or file protection violation
$
```

Correcting: Not Specifying Owner of Directory

To correct this error, use one of the following commands to specify the user's UIC as the owner of the file.

- SET DIRECTORY/OWNER

```
$ SET DIRECTORY/OWNER=[400,7] WORK12:[USER7]
```

or

```
$ SET DIRECTORY/OWNER=[USER7] WORK12:[USER7]
```

- SET FILE/OWNER

```
$ SET FILE/OWNER=[400,7] WORK12:[000000]USER7.DIR
```

or

```
$ SET FILE/OWNER=[USER7] WORK12:[000000]USER7.DIR
```

Error: Incorrectly Locating the User's Directory

If the system manager sets default to SYS\$SYSTEM before creating the user's directory, and does not specify the device when creating the directory, the directory will be located under SYS\$SYSROOT.

Example of the system manager's error:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
.
.
.
$ CREATE/DIRECTORY/OWNER=[11,2] [USER1]
$
```

Example of the user's problem:

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

Correcting: Incorrectly Locating the User's Directory

To correct this error:

- Delete the incorrectly placed directory.

```
$ SET PROTECTION=S:REWD SYS$SYSROOT:[000000]USER1.DIR
$ DELETE SYS$SYSROOT:[000000]USER1.DIR;
$
```

- Create the directory again, specifying the correct device.

```
$ CREATE/DIRECTORY/OWNER=[11,2] DISK$USER:[USER1]
$
OR
$ CREATE/DIRECTORY/OWNER=[USER1] DISK$USER:[USER1]
$
```


Error: Not Setting Up a Disk Quota

If the new user's directory is on a disk where quotas are enabled, and a disk quota has not been established for the user, the following error will occur when the user attempts to create a file:

```
$ EDIT LOGIN.COM
%EDT-F-OPENOUT, error opening DISK$USER:[USER1]LOGIN.JOU; as output
-RMS-E-CRE, ACP file create failed
-SYSTEM-F-EXDISKQUOTA, disk quota exceeded
$
```

Correcting: Not Setting Up a Disk Quota

To correct this error,

1. Run the SYSMAN utility
2. Use the DISKQUOTA command to establish a disk quota for the user

Removing a User Account

Overview

The procedure for removing a user can vary widely from site to site, depending on the site's security policies. Just as in adding an account, when you remove an account you must be careful to remove all aspects of that account.

Topics

In this section the following topics are discussed:

- Preventing the User from Logging in
- Removing Disk Files from the User's Directories
- Removing Entries from the Quota File
- Removing Mail Information
- Remove the UAF record

Preventing the User from Logging In

Preventing Logins

The system manager should prevent logins to the user's account before removing the account itself. This way, if the system manager is interrupted during the procedure, the information about the user is still available.

Preventing the User from Logging In

There are two ways to prevent the user from logging in. Use the Authorize utility for both procedures:

1. Change the user's password

Sample command:

```
UAF> MODIFY USER1/PASSWORD=SAYGOODBYE
```

2. Disable the user's account

Sample command:

```
UAF> MODIFY USER1/FLAGS=DISUSER
```

Removing Disk Files from the User Directories

Removing User Files

Once a user account is disabled, the system manager should remove any files that remain in the user account for several reasons:

- Disk space — Unused files take up space that could be used by active files
- Security — Sensitive files should not be left in an inactive directory
- Accounting — Some sites charge for disk space and backup services

Deciding What to Do with the User's Files

Decide what to do with the user's files before you delete them.

- In some cases, you may delete the entire directory structure from the user's home directory, then delete the home directory.
- There may be a need to archive the files or move them to another directory on the system.

Steps for Removing a User's Files

Delete the user's files and directories from the lowest level up to the top directory level.

1. Locate and examine all subdirectories using the DCL DIRECTORY command.
2. Delete all files in each subdirectory and then delete the subdirectory. Directory files are protected against deletion. Therefore, change the protection code to allow file deletion.
3. Delete all files in the top-level directory and then delete the directory.

Example: Removing a User's Files

The following example shows the removal of a user directory that has two subdirectory levels.

```

❶ $ DIRECTORY DISK$USER:[USER1...]

Directory DISK$USER:[USER1]

LOGIN.COM;1      REPORTS.DIR;1

Total of 2 files.

Directory DISK$USER:[USER1.REPORTS]

DATA.DIR;1      DECEMBER.RPT;1      NOVEMBER.RPT;1      OCTOBER.RPT;1

Total of 4 files.

Directory DISK$USER:[USER1.REPORTS.DATA]

SALES.DAT;1

Total of 1 file.

Grand total of 3 directories, 7 files.
❷ $ DELETE/LOG DISK$USER:[USER1.REPORTS.DATA]*.*;*
%DELETE-I-FILDEL, DISK$USER:[USER1.REPORTS.DATA]SALES.DAT;1 deleted (3 blocks)
❸ $ SET PROTECTION=W:RWED DISK$USER:[USER1.REPORTS]DATA.DIR
❹ $ DELETE/LOG DISK$USER:[USER1.REPORTS]DATA.DIR;
%DELETE-I-FILDEL, DISK$USER:[USER1.REPORTS]DATA.DIR;1 deleted (3 blocks)
❺ $ DELETE/LOG DISK$USER:[USER1.REPORTS]*.*;*
%DELETE-I-FILDEL, DISK$USER:[USER1.REPORTS]DECEMBER.RPT;1 deleted (3 blocks)
%DELETE-I-FILDEL, DISK$USER:[USER1.REPORTS]NOVEMBER.RPT;1 deleted (3 blocks)
%DELETE-I-FILDEL, DISK$USER:[USER1.REPORTS]OCTOBER.RPT;1 deleted (3 blocks)
%DELETE-I-TOTAL, 3 files deleted (9 blocks)
❻ $ SET PROTECTION=W:RWED DISK$USER:[USER1]REPORTS.DIR
❼ $ DELETE/LOG DISK$USER:[USER1]REPORTS.DIR;
%DELETE-I-FILDEL, DISK$USER:[USER1]REPORTS.DIR;1 deleted (3 blocks)
❽ $ DELETE/LOG DISK$USER:[USER1]*.*;*
%DELETE-I-FILDEL, DISK$USER:[USER1]LOGIN.COM;1 deleted (3 blocks)
❾ $ SET PROTECTION=W:RWED DISK$USER:[000000]USER1.DIR
❿ $ DELETE/LOG DISK$USER:[000000]USER1.DIR;
%DELETE-I-FILDEL, DISK$USER:[000000]USER1.DIR;1 deleted (3 blocks)
⓫ $ DIRECTORY DISK$USER:[USER1...]
%DIRECT-E-OPENIN, error opening DISK$USER:[USER1...]*.*;* as input
_RMS-E-DNF, directory not found
-SYSTEM-W-NOSUCHFILE, no such file
$

```

Example Notes: Removing a User's Files

Notes on the Removing a User's Files example include:

- ❶ Locate and examine the subdirectories. There are three directories and seven files.
- ❷ Delete all of the files in the lowest level subdirectory.
- ❸ Set the protection on the lowest level subdirectory so it can be deleted.
- ❹ Delete the lowest level subdirectory.
- ❺ Delete all of the files in the next lowest level subdirectory.
- ❻ Set the protection on the next lowest level subdirectory so it can be deleted.
- ❼ Delete the next lowest level subdirectory.
- ❽ Delete all of the files in the top-level directory.
- ❾ Set the protection on the top-level directory so it can be deleted.
- ❿ Delete the top-level directory.
- ⓫ Verify that the user's directory structure has been deleted.

Removing Entries from the Quota File

Removing Quota Entries

Run the SYSMAN utility to remove disk quota entries under the user account.

Example: Removing Entries from the Quota File

The following example shows how to remove user entries from the Quota file:

```
$ SET PROCESS/PRIVILEGE=OPER
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA REMOVE [11,2] /DEVICE=DISK$USER:
SYSMAN> EXIT
$
```



Note

When running SYSMAN, you can specify the user's identifier instead of the UIC.

Example:

```
SYSMAN> DISKQUOTA REMOVE [USER1] /DEVICE=DISK$USER
SYSMAN>
```

Removing Mail Information

Use the REMOVE command in the Mail utility to remove the user account entry.

Example: Removing Mail Information

The following example shows how to remove a user's mail information:

```
$ MAIL
```

```
MAIL> REMOVE USER1
```

```
MAIL> EXIT
```

```
$
```



Note

You need SYSPRV privileges to remove a mail profile record.

This command removes the personal information from
SYS\$SYSTEM:VMSMAIL_PROFILE.DATA.

Removing the UAF Record

Use the REMOVE command in the Authorize utility to remove a record from the UAF. This permanently prevents the user from logging in.

Example: Removing the UAF Record

The following example shows the removal of the record for USER1 from the User Authorization File.

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF> REMOVE USER1
%UAF-I-REMMSG, record removed from system authorization file
%UAF-I-RDBREMSGU, identifier USER1 value: [000011,000002] removed from rights
database
UAF> EXIT
%UAF-I-DONMSG, system authorization file modified
%UAF-I-NAFNOMODS, no modification made to network proxy database
%UAF-I-RDBDONMSG, rights database modified
$
```

Managing and Monitoring User Processes

Overview

There are several ways in which a system manager can manage and monitor what user processes are allowed to do.

Topics

In this section the following topics are discussed:

- Setting user limits and privileges
- Specifying login command procedures
- Controlling processes
- Using the SYSMAN utility to manage processes on other nodes

Setting User Limits and Privileges

User Limits and Privileges

User limits and privileges are controlled by fields in the user's UAF record.

- Use the ADD command in the Authorize utility to specify initial values for these fields when you create the user UAF record.
- Use the MODIFY command in the Authorize utility to modify these fields for an existing user record.
- To list all of the fields that can be changed, enter the Authorize utility and type HELP MODIFY at the prompt.



Note

By defining a user's privileges, the system manager can control the functions that users are authorized to perform.

- Typically, users have only TMPMBX and NETMBX privileges, which allow them to create files and send mail.
 - The NO prefix removes the privilege from the user.
 - /PRIVILEGES specifies authorized user privileges. These privileges are not necessarily enabled at login.
 - /DEFPRIVILEGES determines which privileges are enabled at login.
 - The keyword NOALL disables all user privileges.
-

Granting or Removing Privileges

Use the MODIFY command in the Authorize utility to modify a user's privileges.

- To grant a privilege, specify /PRIVILEGE=*privilege*.
- To remove a privilege, specify /PRIVILEGE=NO*privilege*.

Example: Granting and Removing Privileges

The following example shows how to grant and remove user privileges:

```
$ SET DEFAULT SYS$SYSTEM
❶ $ RUN AUTHORIZE
❷ UAF> MODIFY USER1 /PRIVILEGE=OPER
    %UAF-I-MDFYMSG, user record(s) updated
❸ UAF> MODIFY USER2 /PRIVILEGE=NOOPER
    %UAF-I-MDFYMSG, user record(s) updated
    UAF> EXIT
    %UAF-I-DONEMSG, system authorization file modified
    %UAF-I-RDBNOMODS, no modifications made to rights data base
$
```

Example Notes: Granting and Removing Privileges

Notes on the Granting and Removing Privileges include:

- ❶ Use the Authorize utility to grant or remove privileges.
- ❷ Grant the OPER (operator) privilege to USER1.
- ❸ Remote the OPER privilege from USER2.

Establishing Limits

By defining user limits, the system manager can limit the system resources for a user.

Use the MODIFY command in the Authorize utility to modify user limits.



Note

Limits are also referred to as quotas in some documentation. Do not confuse limits with disk quotas.

Example: Establishing Limits

The following example sets the value of PRCLM (subprocess creation limit) for USER1 to 12.

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF> MODIFY USER1 /PRCLM=12
%UAF-I-MDFYMSG, user record(s) updated
UAF> EXIT
%UAF-I-DONEMSG, system authorization file modified
%UAF-I-NAFNOMODS, no modifications made to network proxy database
%UAF-I-RDBNOMODS, no modifications made to rights database
$
```

Specifying Login Command Procedures

Login Command Procedures

When a user logs in to an OpenVMS system, one or more command procedures is executed.

- System login command procedure
- User login command procedure

System Login Command Procedure

The system login command procedure establishes elements of a computing environment that will be common to all users.

- This command procedure is executed for each interactive user at login.
- The system logical name SYS\$SYLOGIN points to this command procedure.

User's Login Command Procedure

After the system login command procedure is completed, the user's login command procedure (if there is one) is executed.

- This file can be pointed to by the LGICMD field in the user's UAF record.
- The default file name is LOGIN.COM.
- By default, the file is located in the user's default directory on the user's default device.
- The system manager can specify a different login command procedure for a group of users. Examples of different groups include students in a particular class or members of a project team.

Example: Specifying a Login Command Procedure

In the following example, a group login command procedure is specified for USER1.

```
$ SET DEFAULT SYS$SYSTEM
❶ $ RUN AUTHORIZE
❷ UAF> MODIFY FIN_USER1/LGICMD=WORK6:[FINANCE]GROUP_SETUP.COM
%UAF-I-MDFYMSG, user record(s) updated
❸ UAF> SHOW FIN_USER1

Username:  FIN_USER1           Owner:
Account:   FINANCE             UIC:    [500,1] ([FINANCE,FIN_USER1])
CLI:       DCL                 Tables:  DCLTABLES
Default:   WORK6:[FIN_USER1]
❹ LGICMD:  WORK6:[FINANCE]GROUP_SETUP.COM
.
.
.
UAF>
```

Example Notes: Specifying a Login Command Procedure

Notes on the Specifying a Login Command Procedure example include:

- ❶ Use the Authorize utility to specify a login command procedure for a user.
- ❷ Modify the LGICMD field of the user's UAF record to specify the procedure.
- ❸ Display the user's UAF record to verify the change.
- ❹ The LGICMD field now points to the specified file.

Controlling Processes

Occasionally the system manager needs to manipulate individual processes on the system when:

- A process is in a loop
- A process is using excessive resources and must be stopped
- A user spawns several subprocesses that must share resources and quotas, and one or more of the subprocesses need to be temporarily suspended to free those resources for a more important task

Restricting Processes

There are three ways to control processes on a system:

1. Suspend execution
2. Resume execution
3. Delete the process

Privileges Required to Control Processes

The privileges required to control a given process may be divided into three levels that relate directly to the UIC protection method:

- Owner
- Group
- World

Level	Privileges
Owner	Users can always control their own processes and subprocesses. These processes and subprocesses may have been created with the SPAWN command or the RUN/DETACHED command.
Group	Users with GROUP privilege can control any process that carries the same UIC group identifier as their own.
World	Users with WORLD privilege can control any process on the system. Typically, only the system manager has this privilege.

Suspending a Process

Use the SET PROCESS/SUSPEND command to suspend a process. This action temporarily stops the process's activities. The process remains suspended until another process is resumed or the process is deleted.

Format:

SET PROCESS/SUSPEND/ID=pid

- PID is the process identification number of the process to be suspended.



Note

- GROUP privilege is required to suspend other processes in the same UIC group.
 - WORLD privilege is required to suspend processes outside your UIC group.
-



Note

If the process is in the same UIC group, you can use the following command:

SET PROCESS/SUSPEND process-name

Example: Suspending a Process

In the following example, a process belonging to USER13 is suspended.

```
❶ $ SHOW SYSTEM
OpenVMS T7.0 on node NEAT  23-NOV-1999 09:39:45.11  Uptime 12 21:16:05
  Pid      Process Name  State  Pri      I/O      CPU      Page flts Ph.Mem
❷ 2020071E  USER13        COM    4      1942    0 00:00:56.48      6196  346
.
.
.
❸ $ SET PROCESS/SUSPEND/ID=2020071E
❹ $ SHOW SYSTEM
OpenVMS T7.0 on node NEAT  23-NOV-1999 09:41:35.97  Uptime 12 21:17:56
  Pid      Process Name  State  Pri      I/O      CPU      Page flts Ph.Mem
❺ 2020071E  USER13        SUSP    7      3010    0 00:01:15.81      7291  345
.
.
.
$
```

Example Notes: Suspending a Process

Notes on the Suspending a Process example include:

- ❶ Use the SHOW SYSTEM command to display the process identification code (PID) of the process to be suspended.
- ❷ Note the PID of the process.
- ❸ Use the SET PROCESS/SUSPEND command and specify the PID to suspend the process.
- ❹ Use the SHOW SYSTEM command to verify that the process is suspended.
- ❺ The process state is listed as SUSP, which indicates that the process is suspended.

Resuming a Process

Use the SET PROCESS/RESUME command to resume a process.

Format:

SET PROCESS/RESUME/ID=pid

- PID is the process identification number of the process to be resumed.



Note

- GROUP privilege is required to suspend other processes in the same UIC group.
 - WORLD privilege is required to resume processes outside your UIC group.
-



Note

If the process is in the same UIC group, you can use the following command:

SET PROCESS/RESUME process-name

Example: Resuming a Process

In the following example, a process belonging to USER13 is resumed.

```
❶ $ SHOW SYSTEM
OpenVMS T7.0 on node NEAT  23-NOV-1999 09:39:45.11  Uptime 12 21:16:05
  Pid   Process Name   State Pri    I/O      CPU    Page flts Ph.Mem
❷ 2020071E USER13      SUSP   4    1942    0 00:00:56.48    6196   346
.
.
.
❸ $ SET PROCESS/RESUME/ID=2020071E
❹ $ SHOW SYSTEM
OpenVMS T7.0 on node NEAT  23-NOV-1999 09:41:35.97  Uptime 12 21:17:56
  Pid   Process Name   State Pri    I/O      CPU    Page flts Ph.Mem
❺ 2020071E USER13      COM    7    3010    0 00:01:15.81    7291   345
.
.
.
$
```

Example Notes: Resuming a Process

Notes on the Resuming a Process example include:

- ❶ Use the SHOW SYSTEM command to display the process identification code (PID) of the process to be resumed.
- ❷ The state of the process is SUSP. Note the PID of the process.
- ❸ Use the SET PROCESS/RESUME command and specify the PID to resume the suspended process.
- ❹ Use the SHOW SYSTEM command to verify that the process is resumed.
- ❺ The process state is listed as COM, which indicates that the process is no longer suspended.

Deleting a Process

Use the STOP command to delete a process.

Format:

STOP/ID=pid

- PID is the process identification number of the process to be deleted.



Note

- GROUP privilege is required to stop other processes in the same UIC group.
 - WORLD privilege is required to stop processes outside your UIC group.
-



Note

If the process is in the same UIC group, you can use the following command:

STOP process-name

Example: Deleting a Process

In the following example, a process belonging to USER13 is deleted.

```
❶ $ SHOW SYSTEM
OpenVMS T7.0 on node NEAT 23-NOV-1999 09:39:45.11 Uptime 12 21:16:05
  Pid      Process Name  State Pri      I/O      CPU      Page flts Ph.Mem
❷ 2020071E USER13      COM    4      1942    0 00:00:56.48    6196  346
  .
  .
  .
❸ $ STOP/ID=2020071E
❹ $ SHOW SYSTEM
OpenVMS T7.0 on node NEAT 23-NOV-1999 09:41:35.97 Uptime 12 21:17:56
❺ Pid      Process Name  State Pri      I/O      CPU      Page flts Ph.Mem
  .
  .
  .
$
```

Example Notes: Deleting a Process

Notes on the Deleting a Process example include:

- ❶ Use the SHOW SYSTEM command to display the process identification code (PID) of the process to be deleted.
- ❷ Note the PID of the process.
- ❸ Use the STOP command and specify the PID to delete the process.
- ❹ Use the SHOW SYSTEM command to verify that the process is deleted.
- ❺ The process is not listed in the SHOW SYSTEM output.

Using the SYSMAN Utility to Manage Processes on Other Nodes

SYSMAN Utility

The System Management utility (SYSMAN) enables the system manager to perform system management tasks across nodes and clusters from a local node. The system manager can define a management environment (any node or cluster available through DECnet software) and perform system management tasks in that environment.

Running the SYSMAN Utility

To run the SYSMAN utility, you must have OPER privilege on the local node and authorization for either OPER or SETPRV on the remote node.

Enter the following command to run the SYSMAN utility:

RUN SYS\$SYSTEM:SYSMAN

Exit from the SYSMAN utility by typing EXIT or pressing Ctrl/Z.

Establishing a SYSMAN Environment

Use the SET ENVIRONMENT command in the SYSMAN utility to establish the environment to which subsequent commands will apply. The environment can be:

- Any other node in the cluster
- The entire cluster
- Any node or cluster in the network

This command requires OPER or SETPRV privilege on all nodes in the target environment.

Format:

SET ENVIRONMENT/qualifier(s)

The valid *qualifiers* are:

- /CLUSTER

The default management environment is the local cluster. Specify a nonlocal cluster by naming one cluster member with the /NODE qualifier.

- /NODE=(node1,node2,...)

The node name can be a system name or a cluster alias.

- /USERNAME=username

Specifies the username to be used for access to the other node. If not specified, the current username is used.

Use the SHOW ENVIRONMENT command to display the current environment.



Note

Some DCL commands, such as MOUNT/CLUSTER and SET QUORUM/CLUSTER, operate clusterwide by design. To avoid executing such commands excessively in the SYSMAN utility, define the environment to be a single node within the cluster.

Establishing a SYSMAN Profile

Use the SET PROFILE command to temporarily modify current privileges, default device, and default directory in the current SYSMAN environment. The profile is in effect until you change it, reset the environment, or exit from the SYSMAN utility.

Format:

SET PROFILE/qualifier(s)

The valid *qualifiers* are:

- /DEFAULT=device:[directory]
Specifies the default disk and directory to use in this environment.
- /PRIVILEGES=(priv1,priv2,...)
Specifies the privileges to add to the current privileges. Any enhanced privileges must be authorized.
- /VERIFY
Specifies DCL verification (both procedure and image) for future DO commands.

Use the SHOW PROFILE command to display the current profile.



Note

The initial implementation of SYSMAN was built for clusters. In a cluster, you would not have to set up a profile for yourself.

Executing DCL Commands in the Current Environment

Use the DO command in the SYSMAN utility to execute a DCL command or a command procedure on all nodes in the current environment.

Format:

DO[/output[=file-spec]] command-line

- The optional *output* qualifier is used to record output from the command in a file. The default *file-spec* is SYSMAN.LIS in the current default device and directory.
- The *command-line* parameter specifies a command string to be executed by DCL.

This command requires the same privileges as the DCL command.

Example: Executing Commands on Other Nodes

The following example illustrates SYSMAN utility commands that are used to perform tasks on other nodes.

```

❶ $ SET PROCESS/PRIVILEGE=OPER
❷ $ RUN SYS$SYSTEM:SYSMAN

❸ SYSMAN> SET ENVIRONMENT/CLUSTER/NODE=NEAT
Remote Password:
%SYSMAN-I-ENV, current command environment:
Clusterwide on remote cluster NEAT
Username MGR1          will be used on nonlocal nodes

❹ SYSMAN> SHOW ENVIRONMENT
%SYSMAN-I-ENV, current command environment:
Clusterwide on remote cluster NEAT
Username MGR1          will be used on nonlocal nodes

❺ SYSMAN> SET PROFILE/PRIVILEGE=SYSPRV

❻ SYSMAN> SHOW PROFILE
%SYSMAN-I-DEFDIR, default directory on node NEAT -- $1$DUA1:[MGR1]
%SYSMAN-I-DEFPRIV, process privileges on node NEAT --
SETPRV
TMPMBX
NETMBX
SYSPRV
%SYSMAN-I-DEFDIR, default directory on node TIDY -- $1$DUA1:[MGR1]
%SYSMAN-I-DEFPRIV, process privileges on node TIDY --
SETPRV
TMPMBX
NETMBX
SYSPRV
.
.
.

❼ SYSMAN> DO SHOW SYSTEM
%SYSMAN-I-OUTPUT, command execution on node NEAT
OpenVMS T7.0      on node NEAT   23-NOV-1999 15:00:20.57  Uptime 3
14:04:12
  Pid   Process Name      State Pri    I/O        CPU      Page flts
Ph.Mem
  20200081 SWAPPER        HIB   16      0    0 00:00:29.76      0
0
.
.
.
%SYSMAN-I-OUTPUT, command execution on node TIDY
OpenVMS T7.0      on node TIDY   23-NOV-1999 15:00:08.76  Uptime 3 12:49:29
  Pid   Process Name      State Pri    I/O        CPU      Page flts
Ph.Mem
  40600041 SWAPPER        HIB   16      0    0 00:00:10.97      0
0
.
.
.
❽ SYSMAN> EXIT
$

```

Example Notes: Executing Commands on Other Nodes

Notes on the Executing Commands on Other Nodes example include:

- ❶ OPER privilege is required to run SYSMAN.
- ❷ Invoke the SYSMAN utility.
- ❸ Establish the current environment as the cluster that contains the node NEAT. A password is required, because this is not the local cluster.
- ❹ Display the current environment.
- ❺ Add a privilege to the current profile.
- ❻ Display the current profile. Information will be shown for each node in the environment.
- ❼ Use the DO command to execute a DCL command on each node in the environment. Each command is executed before the SYSMAN utility sends the command to the next node.
- ❽ Exit from the SYSMAN utility.



Note

Each DO command is executed as an independent process; there is no process context retained between DO commands. Therefore, enter all DCL commands in a single command string: you cannot run a program that expects input.

Managing the Use of Disk Space

Overview

System management includes using available disk space efficiently.

Topics

In this section the following topics are discussed:

- Purging old versions of files
- Setting a version limit on directories
- Monitoring disk space
- Managing disk quotas

Purging Old Versions of Files

Old Versions of Files

Purge outdated versions of system log files periodically.

System Files to Purge

Operator log file, SYSS\$MANAGER:OPERATOR.LOG

- A new version of this file is created each time the system is rebooted. Back up and delete outdated versions of files.
- If the operator log file is large, use the REPLY/LOG command to create a new version and back up the previous version.

Error log file, SYS\$ERRORLOG:ERRLOG.SYS

- This file remains on disk until it is explicitly renamed or deleted.
- To free space on the system disk, rename the file on a daily basis, back up the file to a different volume, and then delete the file.

Accounting log file, SYSS\$MANAGER:ACCOUNTNG.DAT

- If accounting activity is enabled, then this file is always increasing in size.
- Create a new accounting file by using the DCL command SET ACCOUNTING/NEW_FILE. Then back up and delete the old file.

Setting a Version Limit on Directories

Version Limits on Directories

To restrict the number of file versions that can be generated in a user's directory, place a version limit on the directory.

- This can be done when the directory is created or for an existing directory.
- A directory version limit specifies the total number of versions for a file.
- When the version limit is exceeded, the earliest version of the file is deleted from the directory without notification to the user.



Note

Be careful about setting a version limit on the master file directory (MFD). Because the system uses the version limit that you set on the MFD on any directory you create beneath the MFD, users might inadvertently lose important data.

Setting a Version Limit When a Directory Is Created

Use the CREATE/DIRECTORY command to set a version limit on a directory at the time it is created. This becomes the default limit for any subdirectories created below this directory.

Format:

CREATE/DIRECTORY/VERSION_LIMIT=*n* *directory-spec*

- The value of *n* is the total number of versions that a file in the specified directory can have. A value of 0 (the default) indicates that the number of versions is limited to 32767, the architectural limit.
- The *directory-spec* is the directory to be created. Wildcard characters are not permitted.



Note

You can use an identifier to specify the owner of the directory:

```
$ CREATE/DIRECTORY/VERSION_LIMIT=3/OWNER=[USER1] WORK6: [USER1]
```

Example: Setting a Version Limit When a Directory Is Created

The following example creates a directory with a version limit of 3.

```
❶ $ CREATE/DIRECTORY/VERSION_LIMIT=3/OWNER=[500,1] WORK6:[FIN_USER1]
❷ $ DIRECTORY/FULL WORK6:[000000]FIN_USER1.DIR

Directory WORK6:[000000]

FIN_USER1.DIR;1          File ID: (3696,14,0)
Size:      1/3          Owner:  [FINANCE,FIN_USER1]
Created: 23-NOV-1999 16:49:46.93
Revised: 23-NOV-1999 16:49:47.17 (1)
Expires: <None specified
Backup: <No backup recorded>
File organization: Sequential
File attributes: Allocation: 3, Extend: 0, Global buffer count: 0
❸ Default version limit: 3, Contiguous, Directory file
.
.
.
$
```

Example Notes: Setting a Version Limit When a Directory Is Created

Notes on the Setting a Version Limit When a Directory is Created example include:

- ❶ Create a directory with a version limit of 3. Be sure to specify the owner of the directory.
- ❷ Use the DIRECTORY/FULL command to display detailed information about the directory.
- ❸ The directory has a version limit of 3. This version limit will be the default for any subdirectories created below this directory.

Setting a Version Limit on an Existing Directory

Use the SET DIRECTORY command to set a version limit on an already existing directory. If you change the version limit for a directory, the new value only affects files created after the change is made.

Format:

SET DIRECTORY/VERSION_LIMIT=*n* *directory-spec*

- The value of *n* is the total number of versions that a file in the specified directory can have. A value of 0 (the default) indicates that the number of versions is limited to 32767, the architectural limit.
- The *directory-spec* is the directory to be modified. Wildcard characters are permitted.

Example: Setting a Version Limit on an Existing Directory

The following example places a version limit of 3 on an existing directory.

```
❶ $ DIRECTORY/FULL DISK$USER:[000000]USER1.DIR

Directory DISK$USER:[000000]

USER1.DIR;1          File ID:  (260,15,0)
Size:      1/3        Owner:    [GRP11,USER1]
Created: 22-NOV-1999 14:26:11.82
Revised: 22-NOV-1999 14:54:53.91 (3)
Expires:  <None specified>
Backup:  23-NOV-1999 17:57:32.02
File organization: Sequential
File attributes: Allocation: 3, Extend: 0, Global buffer count: 0
❷                      No default version limit, Contiguous, Directory file
.
.
.
❸ $ SET DIRECTORY/VERSION_LIMIT=3 DISK$USER:[USER1]
❹ $ DIRECTORY/FULL DISK$USER:[000000]USER1.DIR

Directory DISK$USER:[000000]

USER1.DIR;1          File ID:  (260,15,0)
Size:      1/3        Owner:    [GRP11,USER1]
Created: 22-NOV-1999 14:26:11.82
Revised: 23-NOV-1999 16:35:26.18 (4)
Expires:  <None specified>
Backup:  23-NOV-1999 17:57:32.02
File organization: Sequential
File attributes: Allocation: 3, Extend: 0, Global buffer count: 0
❺                      Default version limit: 3, Contiguous, Directory file
.
.
.
$
```

Example Notes: Setting a Version Limit on an Existing Directory

Notes on the Setting a Version Limit on an Existing Directory example include:

- ❶ Use the DIRECTORY/FULL command to display detailed information about the user's directory.
- ❷ There is no version limit on this directory.
- ❸ Set a version limit of 3 on the directory.
- ❹ Display information about the directory.
- ❺ The directory now has a version limit of 3.



Note

This version limit does not affect files that currently exist in the directory. It will only affect files created after the version limit is assigned.

Monitoring Disk Space

System managers monitor disks to track the amount of space available on the system. Use the DCL command `SHOW DEVICE` to display the number of free blocks on one or more disks.

Example: Displaying Information About a Specific Disk

The following example displays information about the device DUA2:.

```
$ SHOW DEVICE DUA2
```

Device Name	Device Status	Error Count	Volume Label	Free Blocks	Trans Count	Mnt Cnt
\$1\$DUA2: (TIDY)	Mounted	0	USER2	478797	1	11

```
$
```

Example: Displaying Information About a Group of Disks

The following example displays information about devices of the DU class.

```
$ SHOW DEVICE DU
```

Device Name	Device Status	Error Count	Volume Label	Free Blocks	Trans Count	Mnt Cnt
NEAT\$DUA0:	Mounted	0	(remote mount)			1
\$1\$DUA0: (TIDY)	Mounted	0	VMSRL055	120876	260	11
\$1\$DUA1: (TIDY)	Mounted	0	TESTSPACE	475365	1	11
\$1\$DUA2: (TIDY)	Mounted	0	USER2	478797	1	11
\$1\$DUB0: (TIDY)	Online	0				

```
$
```

Actions to Take When Disk Space is Low

If the number of free blocks on a disk is low, take the following actions:

- Purge files, which deletes previous versions of files.
- Move some user directories to another disk. Modify each user's UAF record to reflect the new default device.
- Archive old files that have not been accessed for some time.
- Replace the disk with a device that has a larger capacity.



Note

Monitor all disks daily to make sure they are not too full. Unless a disk contains a very volatile account, it should be no more than 80% full.

Educating Users About Disk Usage

System managers need to educate users about the use of disk space.

- Inform users about the backup schedule. Knowing that a file can be restored helps to persuade users to delete unneeded files.
- Encourage users to purge their directories regularly.
- Educate users to delete unneeded files, such as batch log files and old compiler listings.

Managing Disk Quotas

To establish and control disk quotas, the system manager must do the following:

- Create a quota file on the disk that is to have quotas.
The quota file is automatically enabled when the file is created.
- Establish a disk quota for every UIC user.

Creating and Modifying a Disk Quota File

The quota file, [000000]QUOTA.SYS, controls the disk quotas for the volume on which it is located.

Use the DISKQUOTA command in the SYSMAN utility to create and modify QUOTA.SYS.

To perform this function...	Use this SYSMAN command:
Create and enable a quota file	DISKQUOTA CREATE
Display a quota record	DISKQUOTA SHOW
Add a quota record	DISKQUOTA ADD
Modify a quota record	DISKQUOTA MODIFY
Delete a quota record	DISKQUOTA DELETE
Suspend disk quota enforcement	DISKQUOTA DISABLE
Resume disk quota enforcement	DISKQUOTA ENABLE
Update a quota file on a disk that already has files	DISKQUOTA REBUILD
Remove an entry from a quota file. This procedure requires write (W) access to the quota file.	DISKQUOTA REMOVE



Note

Compaq recommends that you do not create and enable a quota file on the system disk

Default Disk Quotas

Establish default values for quotas and overdrafts by modifying the quota entry for UIC [0,0]. These values will be used for user records unless other values are specified.

Example: Controlling Disk Quotas

The following example creates a disk quota file, establishes default quotas by modifying the record for UIC [0,0], and adds a quota record.

```
❶ SYSMAN> DISKQUOTA CREATE /DEVICE=$1$DUA1:
❷ SYSMAN> DISKQUOTA MODIFY [0,0] /DEVICE=$1$DUA1: -
  _SYSMAN> /PERMQUOTA=2000 /OVERDRAFT=200
❸ SYSMAN> DISKQUOTA ADD [11,7] /DEVICE=$1$DUA1:
❹ SYSMAN> DISKQUOTA SHOW [11,7] /DEVICE=$1$DUA1:

%SYSMAN-I-QUOTA, disk quota statistics on device $1$DUA1: -- Node TIDY
      UIC      Usage  Permanent Quota  Overdraft Limit
[11,7]         0      2000          200

SYSMAN>
```

Example Notes: Controlling Disk Quotas

Notes on the Controlling Disk Quotas example include:

- ❶ Create a quota file on the disk \$1\$DUA1:.
- ❷ Modify the quota record for [0,0] to establish new default quota values for \$1\$DUA1:.
- ❸ Add a quota record for [11,7]. Because no quota values have been specified, the default values will be used.
- ❹ Display the quota entry for [11,7]. The values for [0,0] have been used.

Summary

Concepts

Using the Authorize Utility

- The User Authorization File (UAF) specifies the characteristics of each user process, including the default device and directory, UIC, and limits.
- Use the Authorize utility to manipulate the UAF.
- Use the HELP command of the Authorize utility to get information about AUTHORIZE commands and qualifiers.
- Use the SHOW command of the Authorize utility to display individual UAF records.

Using the Authorize Utility to List User Accounts

- The Authorize utility is used to list user accounts, either at the terminal or in a file.

Adding a User to the System

- Before adding a user to the system, decide on:
 - User name
 - Password
 - User identification code (UIC)
 - Default device
 - Default directory
 - Account name
- Use the Authorize utility to add a record to the User Authorization File.
- Use the SYSMAN utility to establish a disk quota for a user.
- When adding a user to the system, use the CREATE/DIRECTORY command to create a directory that matches the DEFAULT field in the UAF record.
- It is recommended that the system manager test the account before allowing the user to access it.
- ADDUSER.COM is a command procedure that can be used to add users to the system.

Using OpenVMS Management Tools for Windows NT for Adding User Accounts

Includes management of:

- SYSUAF.DAT user authorization file
- TNT\$UADB.DAT file
- RIGHTSLIST.DAT user rights file
- Network proxy database
- Account login directory trees
- User account disk quotas
- OpenVMS Mail VMSMAIL_PROFILE.DATA file

Removing a User Account

- To prevent the user from logging in, use the Authorize utility to either:
 - Change the user's password
 - Disable the user's account
- Once a user has been removed from the system, the system manager should remove any files that remain in the user account.
- Use the SYSMAN utility to remove disk quota entries under the user account.
- Use the REMOVE command in the Mail utility to remove the user account entry.
- Use the REMOVE command in the Authorize utility to remove a record from the UAF.

Managing and Monitoring User Processes

- Use the Authorize utility to modify limits, quotas, and defaults that appear in the UAF.
- The OpenVMS operating system executes up to two command procedures when a user logs in. The first of these is typically a systemwide procedure executed for all users.
- Individual processes can be suspended, resumed, or deleted by the system manager.
- The SYSMAN utility can be used to perform system management tasks on other nodes, such as managing user processes.

Managing the Use of Disk Space

- Outdated versions of system log files should be purged periodically.
- To restrict the number of file versions that can be generated in a user's directory, place a version limit on the directory.
- A directory version limit specifies the total number of versions that a file in the directory can have.
- When the version limit is exceeded, the earliest version of the file is deleted from the directory without notification to the user.
- The system manager should monitor disk space to track the amount of space available on the system.
- To establish and control disk quotas, do the following:
 - Create a quota file on the disk.
 - Establish a disk quota for every UIC that will use that disk.
- The quota file, [000000]QUOTA.SYS, controls the disk quotas for the volume on which it is located.
- Use the DISKQUOTA command in the SYSMAN utility to create and modify the QUOTA.SYS file.

Commands

Using the Authorize Utility

SET DEFAULT SYSS\$SYSTEM
RUN AUTHORIZE

DCL commands to run the Authorize utility.

HELP

Authorize utility command to request help.

SHOW

Authorize utility command to display a UAF record.

Using the Authorize Utility to List User Accounts

LIST

Authorize utility command to write a list of accounts to a file.

SHOW/BRIEF

Authorize utility command to display a list of accounts at the terminal.

Adding a User to the System

ADD

Authorize utility command to add a record to the User Authorization File (UAF).

DISKQUOTA ADD

SYSMAN utility command to add a disk quota record.

CREATE/DIRECTORY/OWNER=uic

DCL command to create a directory.

@SYSS\$EXAMPLES:ADDUSER

DCL command to invoke the command procedure ADDUSER.COM.

Removing a User Account

MODIFY

Authorize utility command to change user passwords or disable accounts.

DIRECTORY

DCL command to identify subdirectories.

SET PROTECTION

DCL command to enable D (delete) access to directory files.

DELETE

DCL command to delete a file.

DISKQUOTA REMOVE

SYSMAN utility command to remove a disk quota record.

REMOVE

MAIL utility command to remove a user profile.

REMOVE

Authorize utility command to remove a record from the User Authorization File (UAF).

Managing and Monitoring User Processes

MODIFY

Authorize utility command to change a record in the User Authorization File (UAF).

MODIFY

AUTHORIZE command to change a record in the User Authorization File (UAF).

SET PROCESS/SUSPEND

DCL command to suspend a process.

SET PROCESS/RESUME

DCL command to resume a process.

STOP

DCL command to stop a process.

SET ENVIRONMENT

SYSMAN utility command to establish the environment to which subsequent commands apply.

Managing the Use of Disk Space

PURGE

DCL command to delete previous versions of files.

CREATE/DIRECTORY/VERSION_LIMIT=n

DCL command to create a directory and specify a version limit.

SET DIRECTORY/VERSION_LIMIT=n

DCL command to specify a version limit on a directory.

SHOW DEVICE

DCL command to display information about devices.

DISKQUOTA CREATE

SYSMAN utility command to create a quota file on a volume.

DISKQUOTA SHOW

SYSMAN utility command to display a quota record.

DISKQUOTA MODIFY

SYSMAN utility command to modify a quota record.

Maintaining Queues

Module 4

About This Module

Introduction

A **queue** is a list of requests for the system to do something in a particular order. As a system manager you will be required to manage and maintain the queues to ensure that all print jobs run as efficiently as possible.

Objectives

To maintain queues, you should be able to:

- Identify the different types of queues.
- Monitor queues and job status.
- Stop a queue.
- Start queues.
- Change the characteristics of a queue.
- Manage jobs in a queue.

OpenVMS Queues

Overview

The system manager or an operator may be responsible for monitoring and controlling the system queues. There are many functions and definitions that must be understood in order to carry out that responsibility.

Topics

In this section the following topics are discussed:

- Generic and execution queues
- Batch and output queues
- The queue manager
- Additional queue managers
- Priority of jobs in a queue

Generic and Execution Queues

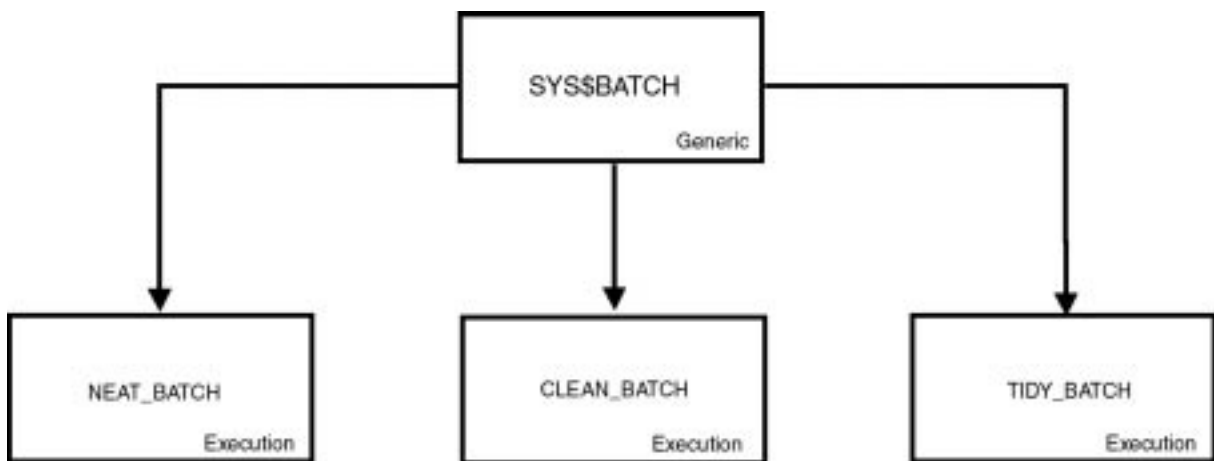
Two Classes of Queues

The OpenVMS operating system provides two general classes of queues:

- Generic — holds jobs until they are transferred to an execution queue for processing
- Execution — accepts batch or print jobs for processing

Figure: Generic and Execution Queues

The following diagram shows the relationship of generic and execution queues.



ZKO-055-000149-01-RGS



Note

A **generic** queue does not process jobs submitted to it. It has associated with it one or more **execution** queues that do the processing.

In the figure, SYSS\$BATCH has been defined as a generic batch queue, and CLEAN_BATCH, NEAT_BATCH, and TIDY_BATCH have been defined as batch execution queues running on nodes CLEAN, NEAT, and TIDY.

Users may submit jobs directly to the execution queues, or they may submit jobs to the generic queue SYSS\$BATCH.

For load balancing, it is better if users submit jobs to the generic queue. The queue manager will direct jobs in a generic queue into the next available execution queue.

Review: Use the SUBMIT command to send a job to a batch queue.

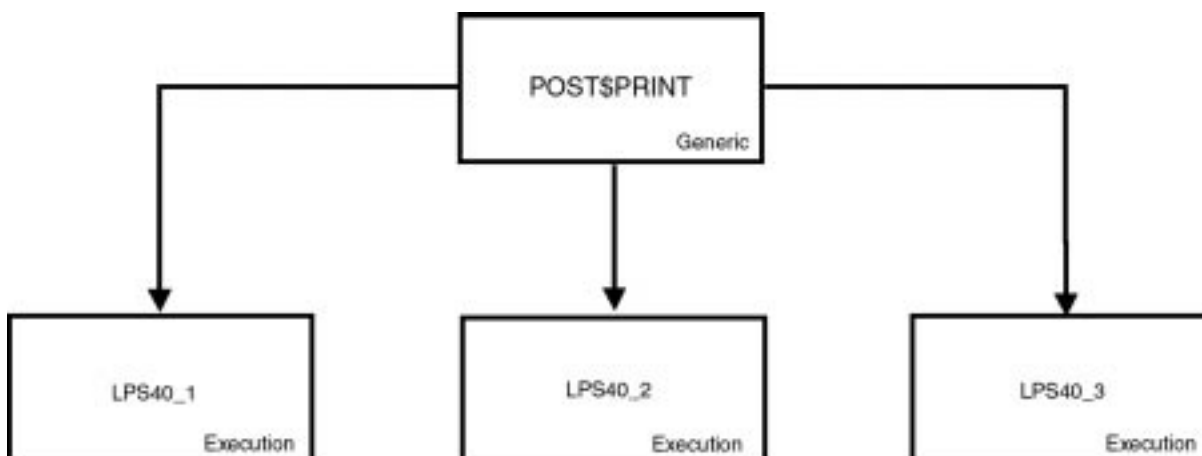
Batch and Output Queues

Execution and generic queues are categorized by the kind of job they accept and the type of device to which the output is directed:

- Batch — directs jobs to batch execution queues
- Output —directs jobs to any of three types of output execution queues:
 1. Print
 2. Terminal
 3. Server

Figure: Output Queues

The following diagram shows output queues going to different printers.



ZKO-055-000149-02-RGS



Note

The difference between a batch and an output queue is in what happens to the job that is submitted. In the case of an output queue, the job is sent to an output device, such as a printer. A batch output queue executes a program on a system node.

In this example, a single generic printer queue is feeding three execution print queues which direct their jobs to individual LPS40 printers.

The queue manager will direct jobs from the generic queue into the next available execution queue.

Queue Manager

The queue manager is a process that runs on a single node of an OpenVMS system. The queue manager:

- Coordinates the jobs on the system as they move between generic and execution queues
- Keeps track of changes made to individual queues
- Maintains a journal file of queue activity

Ordinarily nothing needs to be done to the queue manager once it is running.



Note

By default, an OpenVMS Cluster runs a single queue manager. This queue manager is a single process that runs on one node in the OpenVMS Cluster.

Additional Queue Managers

Running additional queue managers can spread out the queue management load across several OpenVMS Cluster nodes. The distribution of work among queue managers can be on the basis of:

- Print queues versus batch queue
- Queues running on different nodes
- The workload run on your OpenVMS Cluster

The number of queue managers and the workload distribution is up to the system manager.

Queue managers are usually assigned to queues when the queue is initialized. Generic queues and their execution queues must be assigned to the same queue manager.

If a queue manager is not specified when the queue is initialized, the system uses the default queue manager, `SY$QUEUE_MANAGER`.

To assign a different queue manager to a currently existing queue, you must reinitialize the queue.



Note

The OpenVMS operating system requires at least one queue manager to be running. On a node that is NOT part of an OpenVMS Cluster, a single manager is usually sufficient. In an OpenVMS Cluster, more than one queue manager may be required.

Priority of Jobs in a Queue

Priority

Each job that is submitted to a queue has associated with it a priority. Jobs that are waiting in a queue are executed in order of their priority.

- The priority is shown as a number from 0 to 255, with 255 being the highest priority.
- The operating system assigns a default priority of 4 if none is specified.
- It is possible to adjust the priority of a job in a queue.



Note

At this level of skill, the system manager or operator will typically not adjust the priority of jobs in a queue. The discussion here is to introduce the idea, and to point out that the operating system will apply a default value when one is not specified.

There may be occasions when adjusting the priority is necessary, and the method to do this is covered later in this chapter.

Also, if jobs with the same priority are waiting for execution, the one that entered the queue first will execute first.

Another thing that may affect which job goes first is whether or not the execution queue has scheduling enabled. For example, in a queue that has the /SCHEDULE=(SIZE) attribute, small jobs will execute before large jobs.

Example: Priority of a Job in a Queue

The following example shows the use of the SHOW ENTRY command to display characteristics of a job in a batch queue.

```
$ SHOW ENTRY 779/FULL
      Entry Jobname      Username      Blocks Status
      -----
❶      4 TEST          ROUNDS          Holding
      On generic batch queue SYS$BATCH
❷      Submitted 27-SEP-1999 09:15:24.66
      /LOG=$1$DUA1:[ROUNDS]TEST.LOG; /PRIORITY=4
      File: _$1$DUA1:[ROUNDS]TEST.COM;1
```

Example Notes: Priority of a Job in a Queue

Notes on the Priority of a Job example include:

- ❶ The job was submitted with the /AFTER qualifier.
- ❷ The operating system assigns the default priority of 4.



Note

Additional notes on example:

- /KEEP means the log file from the batch job will not be deleted.
- /NOPRINT means the log file will not be sent to a printer.

The last part of the display names the command procedure(s) to be executed by this batch job.

Monitoring Queues and Jobs

Overview

It is a good idea to periodically check system queues with a `SHOW QUEUE` command. To see if a queue has stopped unexpectedly, or has stalled (possibly due to a device problem). Corrective action can then be taken before users are affected.

You can monitor the status of an entire queue or of individual jobs within a queue. The amount of information displayed depends on your privileges and queue ownership rights.

Topics

In this section the following topics are discussed:

- Monitoring queue status
- Tailoring the `SHOW QUEUE` command
- Monitoring job status

Monitoring Queue Status

You should periodically monitor the status of the queues on the system. Things to look for include:

- Queues that have stopped unexpectedly
- Batch jobs that appear to be looping
- Execution queues that have stalled due to a hardware problem



Note

This section talks about the use of the SHOW command to monitor queues and jobs on the system. The SHOW command gives a good picture of the activity on the system, and can be used to see problems before users become aware of them.

SHOW QUEUE Command

Use the DCL command SHOW QUEUE to monitor the status of entire queues.

Format:

SHOW QUEUE [/qualifier...] [queue-name]

- If you do not specify a queue-name, the system displays information on all queues
- SHOW QUEUE qualifiers allow you to select the type and amount of information you want to display

Example: Output from the SHOW QUEUE Command

The following example shows output from a SHOW QUEUE command on a system where there are several queues.

```
$ SHOW QUEUE
❶ Generic printer queue ANSI_20

Generic printer queue ANSI_40

❷ Batch queue MESSY_BATCH, idle, on MESSY::

Batch queue NEAT_BATCH, idle, on NEAT::

Batch queue TIDY_BATCH, idle, on TIDY::

Printer queue LPS20$CRIER, idle, on NEAT::CRIER, mounted
form LPS$$FORM
(stock=DEFAULT)

❸ Printer queue LPS40$JAMED, stopped, on NEAT::JAMED,
mounted form LPS$$FORM (stock=DEFAULT)

Printer queue LPS40$SCDTST, idle, on NEAT::SCDTST, mounted
form LPS$$FORM
(stock=DEFAULT)

Generic printer queue PS_20

Generic printer queue PS_40

Generic batch queue SYS$BATCH

Generic printer queue SYS$PRINT

$
```



Note

For the moment we will talk only about showing information about queues; that is why the /ALL qualifier is not described in the next few pages.

Example Notes: Output from the SHOW QUEUE Command

Notes on the Output from the SHOW QUEUE example include:

- ❶ These generic printer queues feed an LPS20 and an LPS40 printer.
- ❷ A queue is idle when it is waiting for jobs.
- ❸ This queue was stopped due to a hardware problem with the printer.

Tailoring the SHOW QUEUE Command

SHOW QUEUE Command

The SHOW QUEUE command takes a number of qualifiers that allow you to display selected subsets of all of the queues on a system.

SHOW QUEUE Qualifiers

To display...	Use this command:
All of the queues on a system	SHOW QUEUE
All batch queues	SHOW QUEUE /BATCH
All generic queues	SHOW QUEUE /GENERIC
All printer, terminal, and server queues	SHOW QUEUE /DEVICE

Example: Displaying Generic Queues

The following example shows the use of the /GENERIC qualifier, which displays only the generic queues on a system.

```
$ SHOW QUEUE/GENERIC
```

```
Generic printer queue ANSI_20
```

```
Generic printer queue ANSI_40
```

```
Generic printer queue PS_20
```

```
Generic printer queue PS_40
```

```
Generic batch queue SYS$BATCH
```

```
Generic printer queue SYS$PRINT
```

```
$
```

Using the /FULL and /ALL

The /FULL and /ALL qualifiers are used to get more information on queues.

To display...	Use the command...
Detailed information about queue attributes	SHOW QUEUE /FULL
Information about jobs in the queue	SHOW QUEUE /ALL

Using Wildcards to Display Queue Information

You can show information about a single queue or a group of related queues by specifying either an explicit queue name or a wildcarded queue name.

```
$ SHOW QUEUE LPS*
```

```
Printer queue LPS20$CRIER, idle, on NEAT::CRIER, mounted form  
LPSS$$FORM  
  (stock=DEFAULT)
```

```
Printer queue LPS20$FORPUB, idle, on NEAT::FORPUB, mounted form  
LPSS$$FORM  
  (stock=DEFAULT)
```

```
Printer queue LPS40$JAMED, idle, on NEAT::JAMED, mounted form  
LPSS$$FORM  
  (stock=DEFAULT)
```

```
Printer queue LPS40$SCDTST, idle, on NEAT::SCDTST, mounted form  
LPSS$$FORM  
  (stock=DEFAULT)  
$
```

Identifying SHOW QUEUE Status Codes

The following table explains what action is taking place when you see specific status codes returned from the SHOW QUEUE command.

The status code...	Means that...
Aligning	The queue manager is processing a START/QUEUE/ALIGN command.
Autostart inactive	The queue was stopped and needs to be activated.
Available	The queue is processing at least one job but is capable of processing additional concurrent jobs.
Busy	The queue cannot process additional jobs because of one or more jobs in progress.
Closed	The queue is closed and will not accept new jobs until it is set open.
Device unavailable	The device to which the print queue is assigned is not available.
Idle	The queue is waiting for jobs.
Paused	A STOP/QUEUE command has been executed.
Pausing	The queue manager is processing a STOP/QUEUE command.
Remote	The queue is assigned to a physical device that is not connected to the local system.
Resuming	The queue manager is processing a START/QUEUE command on a paused queue.
Server	The queue processing is directed to a server symbiont.
Stalled	Print processing is temporarily halted due to a device-related problem.
Starting	The queue has been started, but is not active (ready to print jobs).
Stop pending	The queue will be stopped when current jobs have finished executing.
Stopped	A STOP/QUEUE command specified with a /NEXT, /REQUEUE, or /RESET qualifier has been executed.
Stopping	The queue manager is processing a STOP/QUEUE command specified with a /NEXT, /REQUEUE, or /RESET qualifier.

Monitoring Job Status

You can monitor the status of jobs belonging to a particular user, or individual print or batch jobs.

SHOW ENTRY Command

Use the DCL SHOW ENTRY command to monitor the status of individual jobs.

Format:

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

- Various *qualifiers* allow you to filter the information displayed about jobs on the system.



Note

You must have OPER privilege or Read/Write/Execute privileges to the queue to display jobs owned by other users.



Note

If you do not specify an entry number, the system displays all jobs owned by you (or by the user specified with the /USER qualifier).

Specifying Job Information to Display

Use the following commands to specify job information you want displayed.

To display information about...	Use the command...
All jobs owned by you	SHOW ENTRY
Job number 228	SHOW ENTRY 228
All jobs owned by user JONES	SHOW ENTRY /USER=JONES
All jobs that begin with the letter B	SHOW ENTRY B*

Example: Displaying Job Status Information

The following example displays information about all jobs owned by USER1 that begin with the letter A.

```
$ SHOW ENTRY/USER=USER1 A*
Entry  Jobname      Username      Blocks Status
-----
  777   AGEN_BATCH    USER1        Holding until 17-SEP-1999
14:00
      On generic batch queue SYS$BATCH
```

Job Status Codes

Jobs, like queues, have codes associated with their status. The table below lists job status codes returned by the SHOW ENTRY command.

The status code...	Means that...
Aborting	Executing job is terminating prior to normal completion.
Executing	Job is executing from a batch queue.
Holding	Job is being held until explicitly released.
Holding until	Job is being held until a specified time.
Pending	Job is in a wait state, typically waiting to be processed.
Printing	Job is executing from a printer or terminal execution queue.
Processing	Job is executing from a server queue.
Retained on completion	Job remains in the queue upon completion.
Retained on error	Job remains in the queue upon encountering an error.
Stalled	Physical device on which the job was executing is stalled.
Starting	The job is beginning to be processed.
Suspended	The job stopped during processing, but should continue when the cause of the suspended state is resolved.
Waiting	The job has been refused (Print queues only).



Note

The STALLED status is probably the most important to the novice system manager or operator, since it indicates that a physical device, such as a printer, has a problem.

Stopping a Queue

Overview

Sometimes it will be necessary to stop a queue in progress. Using OpenVMS, you can stop, pause or close the queue depending on the circumstances.

Topics

In this section the following topics are discussed:

- Ways to stop a queue
- Pausing a queue
- Orderly shutdown of a queue
- Immediate shutdown of a queue
- Closing a queue
- Shutting down all queues on a single node
- Shutting down queues on all nodes in an OpenVMS Cluster

Stopping a Queue

Queue management includes:

- Pausing a queue
- Stopping a queue in an orderly manner
- Stopping a queue immediately
- Closing a queue
- Stopping all queues on a single node
- Stopping all queues in an OpenVMS Cluster

In each of these cases except closing a queue, the STOP/QUEUE command is used.

Format:

STOP/QUEUE [/qualifiers] queue_name

Pausing a Queue

In some situations the best method of deleting an executing job from a queue is to pause the queue, delete the job, then restart the queue.



Note

On some printers, deleting an executing job will cause the printer to print subsequent pages out of order.

STOP/QUEUE Command

To pause a queue, use the STOP/QUEUE command. This command requires OPER privilege.

This command:

- Pauses the queue
- Suspends execution of the current job

Format:

STOP/QUEUE queue-name

Example: Pausing a Queue

The following example shows how to pause a queue:

```
❶ $ SUBMIT/QUEUE=CLEAN_BATCH LOOPS.COM
    Job LOOPS (queue CLEAN_BATCH, entry 988) started on
    CLEAN_BATCH

❷ $ STOP/QUEUE CLEAN_BATCH

    $ SHOW QUEUE CLEAN_BATCH
❸ Batch queue CLEAN_BATCH, paused, on CLEAN::
```

Entry	Jobname	Username	Status
-----	-----	-----	-----
❹ LOOPS	USER5	Suspended	

```
❺ $ SUBMIT/QUEUE=CLEAN_BATCH LOOPS.COM
    Job LOOPS (queue CLEAN_BATCH, entry 989) pending
    pending status caused by queue paused state

$
```

Example Notes: Pausing a Queue

Notes on the Pausing a Queue example include:

- ❶ LOOPS.COM is a DCL command procedure that loops indefinitely.
- ❷ CLEAN_BATCH is paused using the STOP command.
- ❸ The queue is paused.
- ❹ The job is suspended; to continue execution, the queue must be started.
- ❺ Further submissions to the paused queue will not execute until the queue is started.

Orderly Shutdown of a Queue

To perform an orderly shutdown of a queue, use the STOP/QUEUE/NEXT command. This command requires OPER privilege.

This command stops the queue after the currently executing jobs are completed.

Format:

STOP/QUEUE/NEXT queue-name

Example: Orderly Shutdown of a Queue

The following example shows how to perform an orderly queue shutdown:

❶ \$ STOP/QUEUE/NEXT CLEAN_BATCH

\$ SHOW QUEUE CLEAN_BATCH

❷ Batch queue CLEAN_BATCH, available, stop pending, on CLEAN::

Entry	Jobname	Username	Status
-----	-----	-----	-----
988	LOOPS	USER3	Executing

❸ \$ DELETE/ENTRY=988

\$ SHOW QUEUE CLEAN_BATCH

❹ Batch queue CLEAN_BATCH, stopped, on CLEAN::

\$

Example Notes: Orderly Shutdown of a Queue

Notes on the Orderly Shutdown of a Queue example include:

- ❶ Cause an orderly shutdown of the queue.
- ❷ The queue is in a *stop pending* state because a job is currently executing.
- ❸ The job LOOPS is deleted from the queue; normally the job would end on its own.
- ❹ Once the currently executing job is complete, the queue stops.



Note

This is the best method to use if the circumstances allow it. In the example, the DELETE/ENTRY command is used to stop execution of LOOPS.COM, which loops indefinitely. In normal situations, the DELETE command would not be necessary; the queue would be stopped after the currently executing job is finished.

Immediate Shutdown of a Queue

To perform the immediate shutdown of a queue, use the STOP/QUEUE/RESET command. This command requires OPER privilege.

This command:

- Abruptly stops the queue
- Stops all currently executing jobs in the queue

Format:

STOP/QUEUE/RESET queue-name



Note

Jobs that were pending in the queue before the shutdown will still be pending when the queue is restarted.

Example: Immediate Shutdown of a Queue

The following example shows how to perform an immediate shutdown of a queue.

❶ \$ SUBMIT/QUEUE=CLEAN_BATCH LOOPS.COM

Job LOOPS (queue CLEAN_BATCH, entry 990) started on CLEAN_BATCH

\$ SHOW QUEUE CLEAN_BATCH

Batch queue CLEAN_BATCH, busy, on CLEAN::

Entry	Jobname	Username	Status
-----	-----	-----	-----
990	LOOPS	USER5	Executing

❷ \$ STOP/QUEUE/RESET CLEAN_BATCH

❸ \$ SHOW QUEUE CLEAN_BATCH

Batch queue CLEAN_BATCH, stopped, on CLEAN::

\$

Example Notes: Immediate Shutdown of a Queue

Notes on the Immediate Shutdown of a Queue example include:

- ❶ LOOPS.COM is a DCL command procedure that loops indefinitely.
- ❷ Shut down the queue abruptly.
- ❸ The job that was executing is immediately terminated.

Closing a Queue

To close a queue, use the SET QUEUE/CLOSE command. This command requires OPER privilege.

This command prevents jobs from being entered in the queue through the commands PRINT or SUBMIT, or as a result of requeue operations.

Format:

SET QUEUE/CLOSE queue-name

Note: To allow jobs to be entered into the queue again, issue the SET QUEUE/OPEN command.



Note

Other methods of stopping a queue, such as STOP/RESET and STOP/NEXT, allow the stopped queue to continue accepting jobs, which are put into a **pending** state. When the queue is restarted, the jobs that have been pending begin to execute.

Closing a queue causes the queue to refuse all jobs submitted to it.

Example: Closing a Queue

The following example shows how to close a queue.

❶ \$ SUBMIT/QUEUE=CLEAN_BATCH LOOPS.COM

Job LOOPS (queue CLEAN_BATCH, entry 991) started on CLEAN_BATCH

\$ SHOW QUEUE CLEAN_BATCH

Batch queue CLEAN_BATCH, busy, on CLEAN::

Entry	Jobname	Username	Status
-----	-----	-----	-----
991	LOOPS	USER5	Executing

❷ \$ SET QUEUE/CLOSE CLEAN_BATCH

\$ SHOW QUEUE CLEAN_BATCH

❸ Batch queue CLEAN_BATCH, busy, closed, on CLEAN::

Entry	Jobname	Username	Status
-----	-----	-----	-----
991	LOOPS	USER5	Executing

❹ \$ SUBMIT/QUEUE=CLEAN_BATCH LOOPS.COM

%SUBMIT-F-CREJOB, error creating job

-JBC-E-QUE_CLOSED, queue closed, jobs not accepted

\$

Example Notes: Closing a Queue

Notes on the Closing a Queue example include:

- ❶ LOOPS.COM is a DCL command procedure that loops indefinitely.
- ❷ Close the queue.
- ❸ LOOPS is still executing, however the queue status is **closed**.
- ❹ Further submissions to the queue are not accepted.

Shutting Down All Queues on a Single Node

To shut down all of the queues on a single node in an orderly manner, use the STOP/QUEUE/ON_NODE command. This command requires OPER and SYSNAM privileges.

This command performs an orderly shutdown of all queues on the specified node after the currently executing jobs are completed.

If you do not specify node_name, queues on the current node are shut down.

Format:

STOP/QUEUE/ON_NODE[=node_name]



Note

STOP/QUEUE/ON_NODE is for OpenVMS V5.5 and later.

There is no equivalent command to *start* all of the queues on a single node.

Example: Shutting Down All Queues on a Single Node

The following example shows how to shut down all queues on a single node.

```
❶ $ STOP/QUEUE/ON_NODE=CLEAN

$ SHOW QUEUE

Generic printer queue ANSI_20

Generic printer queue ANSI_40

Batch queue NEAT_BATCH, idle, on NEAT::

Batch queue TIDY_BATCH, idle, on TIDY::

❷ Batch queue CLEAN_BATCH, stopped, on CLEAN::

❷ Batch queue CLEAN_BIG, stopped, on CLEAN::

❷ Batch queue CLEAN_SMALL, stopped, on CLEAN::
$
```

Example Notes: Shutting Down All Queues on a Single Node

Notes on the Shutting Down All Queues on a Single Node example include:

- ❶ Shut down all queues on node CLEAN.
- ❷ Only queues on CLEAN are stopped.

Shutting Down Queues on All Nodes in an OpenVMS Cluster

To shut down the queues on all nodes in an OpenVMS Cluster, use the STOP/QUEUE/MANAGER/CLUSTER command. This command requires OPER and SYSNAM privileges.

Format:

STOP/QUEUE/MANAGER/CLUSTER

Using /NAME_OF_MANAGER

If your OpenVMS Cluster has additional queue managers, they must also be shut down. This is done by adding the /NAME_OF_MANAGER qualifier to the command.

Format:

STOP/QUEUE/MANAGER/CLUSTER/NAME_OF_MANAGER=name



Note

To shut down all queues on an OpenVMS Cluster system, shut down the queue manager.

The queue manager keeps track of the state of all queues on the system. When the queue manager is shut down and then restarted, it will reset the queues to the state that they were in prior to the shutdown.

In other words, a queue that was stopped when the queue manager was shut down will still be stopped when the queue manager is restarted.

STOP/QUEUE/MANAGER/CLUSTER is for OpenVMS V5.5 and later.

If the queue manager is stopped at the time a node is shut down, the queue manager will not automatically restart when the node is rebooted. An explicit START/QUEUE/MANAGER command must be issued.

Example: Shutting Down Queues on All Nodes in an OpenVMS Cluster

The following example show how to shut down queues on all nodes on a VMSCluster.

```
$ STOP/QUEUE/MANAGER/CLUSTER
```

```
$ SUBMIT/QUEUE=CLEAN_BATCH LOOPS.COM
```

```
%SUBMIT-F-CREJOB, error creating job
```

```
-JBC-E-JOBQUEDIS, system job queue manager is not running
```

```
$ SHOW QUEUE
```

```
%JBC-E-JOBQUEDIS, system job queue manager is not running
```

```
$
```


Starting Queues

Overview

If the queue manager has been stopped, it can be restarted with the START/QUEUE/MANAGER command. If a queue has been stopped, it can be restarted with the START/QUEUE command.

Topics

In this section the following topics are discussed:

- Starting the queue manager
- Starting additional queue managers
- Starting a queue

Starting the Queue Manager

To start the QUEUE_MANAGER process, use the START/QUEUE/MANAGER command. This command requires OPER and SYSNAM privileges.

The first time this command is entered, it is stored in the queue database.

On subsequent reboots, the queue manager starts automatically.

Format:

START/QUEUE/MANAGER [/ON=(node,...)]

When to Start the Queue Manager

Because the START/QUEUE/MANAGER command is stored in the queue database and executed automatically at boot time, you no longer need to place the START command in a startup command procedure such as SYSTARTUP_VMS.COM. The queue manager starts when the JOB_CONTROL process starts.

If the queue manager stops when a system shuts down, however, it will not automatically restart when the system is rebooted; you must enter an explicit START/QUEUE/MANAGER command.

Specifying Nodes on Which to Run the Queue Manager

In a clustered environment, the `START/QUEUE/MANAGER/ON=(node,...)` command has two functions:

- Create a list of nodes, in order of preference, that the queue manager may run on.
- Force failover of the queue manager to a specified node.

Sample command:

```
$ START/QUEUE/MANAGER/ON=(NEAT, TIDY, *)
```

- Attempt to start the queue manager on node NEAT. If node NEAT is not available, node TIDY is tried.
- The asterisk indicates that any other available node in the cluster is eligible to run the queue manager after the specifically named nodes have been tried.
- Specifying a single node name in the list will force the queue manager to move to the specified node if it is available.



Note

In a clustered environment, always specify the `/ON=(node,...)` qualifier. If no node list has been specified, and the node on which `QUEUE_MANAGER` is running leaves the cluster, no new queue manager is started.

Typically the boot node is the first node listed.

Starting Additional Queue Managers

To start additional queue manager processes, use the `START/QUEUE/MANAGER/NAME_OF_MANAGER=name` command. This command requires OPER and SYSNAM privileges.

The first time you enter this command, it is stored in the queue database.

On subsequent reboots, this queue manager starts automatically.

Format:

START/QUEUE/MANAGER/NAME_OF_MANAGER=name
[/ON=(node,...)]

- The value of *name* is the queue manager's process name. Note: Process names are 15 characters in length. Any name you select will be truncated to the appropriate length.

When to Start the Queue Manager

Because the `START/QUEUE/MANAGER/NAME_OF_MANAGER=name` command is stored in the queue database and executed automatically at boot time, you no longer need to place the `START` command in a startup command procedure such as `SYSTARTUP_VMS.COM`. The queue manager starts when the `JOB_CONTROL` process starts.

If this queue manager is stopped when a system is shut down, however, it will not automatically restart when the system is rebooted; an explicit `START/QUEUE/MANAGER/NAME_OF_MANAGER=name` command must be entered.

Specifying Nodes on Which to Run the Queue Manager

In a clustered environment, the `START/QUEUE/MANAGER/NAME_OF_MANAGER=name/ON=(node,...)` command has several functions:

- Start an additional queue manager.
- Create a list of nodes, in order of preference, that the additional queue manager may run on.
- Force failover of the additional queue manager to a specified node.

Sample command:

```
$  
START/QUEUE/MANAGER/NAME_OF_MANAGER=FREDS_Q_MGR/ON=( CLEAN, * )
```

- Attempt to start the queue manager, `FREDS_Q_MGR`, on node `CLEAN`.
- The asterisk indicates that any other available node in the cluster is eligible to run the queue manager after the specifically named nodes have been tried.
- Specifying a single node name in the list forces the queue manager to move to the specified node if it is available.



Note

In a clustered environment, you should always specify the `/ON=(node,...)` qualifier. If no node list has been specified, and the node on which the additional queue manager, `FREDS_Q_MGR`, is running leaves the cluster, no new queue manager is started.

Starting a Queue

To start an existing queue, use the START/QUEUE command. This command requires OPER privilege.

Format:

START/QUEUE [/qualifiers] queue-name[:]



Note

Queue commands such as START and STOP are stored in the system queue database. For example, when a queue is started, the information is stored in the queue database, and the queue is automatically started when the system reboots.

This is not the same as an autostart queue. Setting a queue to be an autostart queue is a way of guaranteeing that the queue will start when an ENABLE AUTOSTART command is entered, which is typically done in SYSTARTUP_VMS.COM.

Example: Starting a Queue

The following example shows how to start a queue.

```
❶$ SHOW QUEUE ANSI_20
   Generic printer queue ANSI_20, stopped
$
❷$ START/QUEUE ANSI_20
$
❸$ SHOW QUEUE ANSI_20
   Generic printer queue ANSI_20
$
❹$ SUBMIT/QUEUE=TIDY_BATCH NUMBERS.COM
   Job NUMBERS(queue TIDY_BATCH, entry 930) pending
       pending status caused by queue stopped state
❺$ SHOW QUEUE TIDY_BATCH
   Batch queue TIDY_BATCH, stopped, autostart inactive, on
TIDY::
```

Entry	Jobname	Username	Status
-----	-----	-----	-----
930	NUMBERS	SYSTEM	Pending

```
❻$ START/QUEUE TIDY_BATCH
❼$ SHOW QUEUE TIDY_BATCH
   Batch queue TIDY_BATCH, available, on TIDY::
```

Entry	Jobname	Username	Status
-----	-----	-----	-----
930	NUMBERS	SYSTEM	Executing

\$

Example Notes: Starting a Queue

Notes on the Starting a Queue example include:

- ❶ The output from the SHOW QUEUE command indicates that this generic print queue is stopped.
- ❷ Start the print queue.
- ❸ The print queue is no longer stopped.
- ❹ Submit a job to a batch queue that has been stopped.
- ❺ The job is pending because the batch queue is stopped.
- ❻ Start the batch queue.
- ❼ The job has started to execute.

Changing the Characteristics of a Queue

Overview

Queue management may require the system manager to change certain characteristics of queues, including size restrictions and scheduling considerations.

Topics

In this section the following topics are discussed:

- Changing queue characteristics
- Limiting job sizes

Changing Queue Characteristics

To change the characteristics of a queue, use the SET QUEUE command. This command requires OPER privilege.

Format:

SET QUEUE/qualifier[...] queue-name

Sample SET QUEUE Qualifiers

The following table lists some commonly used qualifiers to the SET QUEUE command.

To change this characteristic...	Use this qualifier...
Minimum size of jobs allowed in the queue	/BLOCK_LIMIT=(lowlim, "")
Maximum size of jobs allowed in the queue	/BLOCK_LIMIT=uplim
Order of execution; short jobs go before larger jobs	/SCHEDULE=SIZE
Order of execution; jobs execute in the order that they arrive subject to them having the same priority	/SCHEDULE=NOSIZE

Characteristics of Queues

Use the HELP SET QUEUE command to list all of the queue characteristics that can be changed.



Note

The SET QUEUE/BLOCK_LIMIT command limits the size of print jobs that can be processed on an output execution queue. This qualifier allows you to reserve certain printers for certain size jobs. You must specify at least one of the parameters.

The LOWLIM parameter is a decimal number referring to the minimum number of blocks accepted by the queue for a print job.

If a print job is submitted that contains fewer blocks than the LOWLIM value, the job remains pending until the block limit for the queue is changed. After the block limit for the queue is decreased sufficiently, the job is processed.

The UPLIM parameter is a decimal number referring to the maximum number of blocks that can be accepted by the queue for a print job.

If a print job is submitted that exceeds this value, the job remains pending until the block limit for the queue is changed. After the block limit for the queue is increased sufficiently, the job is processed.

Limiting Job Sizes

Job Size

Limiting print job sizes is especially important when trying to balance the load between printers that differ in performance, but otherwise have similar queue characteristics.



Note

The only time the operating system pays attention to job size is at the time a job is transferred from a generic to an execution queue. If a user types \$ PRINT/QUEUE=SMALL X and X.LIS is over 100 blocks and the queue SMALL has a limit of 100 blocks, the job will just sit there.

Limiting Job Sizes

Use the SET QUEUE/BLOCK_LIMIT command to limit the size of print jobs that can be processed on an output execution queue. This command requires OPER privilege.

Format:

SET QUEUE/BLOCK_LIMIT=(*lowlim*,*uplim*) queue-name

- The *lowlim* parameter specifies the minimum number of blocks that are accepted by the queue for a print job.
If a print job is submitted that contains fewer blocks than the *lowlim* value, the job remains pending until the block limit for the queue is changed.
- The *uplim* parameter specifies the maximum number of blocks that are accepted by the queue for a print job.
If a print job is submitted that exceeds this value, the job remains pending until the block limit for the queue is changed.
- If you specify only an upper limit for jobs, you can omit the parentheses. For example, /BLOCK_LIMIT=1000 means that only jobs with 1000 blocks or less are processed in the queue.
- To specify only a lower job limit, you must use two consecutive double quotation marks (""") to indicate the upper limit. For example, /BLOCK_LIMIT=(500, "") means any job with 500 or more blocks is processed in the queue.
- You can specify both a lower and upper limit. For example, /BLOCK_LIMIT=(200,2000) means that jobs with less than 200 blocks or more than 2000 blocks are not processed in the queue.

The /NOBLOCK_LIMIT qualifier cancels the previous block limit setting for that queue.

Scheduling Print Jobs by Size

Use the SET QUEUE/SCHEDULE command to specify whether pending jobs in an output queue are scheduled for printing based on the size of the job. This command requires OPER privilege.



Note

If you enter this command while there are pending jobs in any queue, the effect on future jobs is unpredictable.

Format:

SET QUEUE/SCHEDULE=[NO]SIZE queue-name[:]

- When the /SCHEDULE=SIZE qualifier is in effect, shorter jobs print before longer ones.
- When the /SCHEDULE=NOSIZE qualifier is in effect, jobs are printed in the order they were submitted, regardless of size.

Example: Changing Queue Characteristics

The following example shows the use of the SET QUEUE command to change the characteristics of a queue.

```
$ SHOW QUEUE LPS40$SCDTST/FULL
Printer queue LPS40$SCDTST, idle, on NEAT::SCDTST, mounted
form LPS$$$FORM
(stock=DEFAULT)
/AUTOSTART_ON=(NEAT::SCDTST,*::SCDTST) /BASE_PRIORITY=4
/DEFAULT=(FORM=LPS$$$FORM (stock=DEFAULT))
/NOENABLE_GENERIC
/LIBRARY=LPS$DEVCTL Lowercase /OWNER=[SYSTEM]
/PROCESSOR=LPS$SMB
❶ /PROTECTION=(S:M,O:D,G:R,W:S) /SCHEDULE=(NOSIZE)
/SEPARATE=(FLAG,
  RESET=(LPS$$EOJ),TRAILER)

❷ $ SET QUEUE/BLOCK_LIMIT=500 LPS40$SCDTST

❸ $ SET QUEUE/SCHEDULE=SIZE LPS40$SCDTST

$ SHOW QUEUE LPS40$SCDTST/FULL
Printer queue LPS40$SCDTST, idle, on NEAT::SCDTST, mounted
form LPS$$$FORM
(stock=DEFAULT)
❹ /AUTOSTART_ON=(NEAT::SCDTST,*::SCDTST) /BASE_PRIORITY=4
/BLOCK_LIMIT=500
/DEFAULT=(FORM=LPS$$$FORM (stock=DEFAULT))
/NOENABLE_GENERIC
/LIBRARY=LPS$DEVCTL Lowercase /OWNER=[SYSTEM]
/PROCESSOR=LPS$SMB
/PROTECTION=(S:M,O:D,G:R,W:S)
/SEPARATE=(FLAG,RESET=(LPS$$EOJ),TRAILER)
```

Example Notes: Changing Queue Characteristics

Notes on the Changing Queue Characteristics example include:

- ❶ By default, the queue executes jobs in the order that they arrive (/SCHEDULE=(NOSIZE)).
- ❷ Set the queue with an upper block size limit of 500.
- ❸ Set the queue to execute jobs in order of their size.
- ❹ The queue now has a block limit.



Note

If you change the SCHEDULE characteristic while there are pending jobs in any queue, the effect on future jobs is unpredictable.

Managing Jobs in a Queue

Overview

Periodically monitor print and batch jobs that execute on the system. Some of the problems that may occur include:

- Print jobs set to a printer with no paper
- Large print job executing while smaller, more important jobs wait
- Batch jobs running that are scheduled incorrectly

In situations such as these, the jobs should be stopped or requeued.

Topics

In this section the following topics are discussed:

- Aborting and requeuing jobs on output queues
- Moving a job to a different queue
- Merging queues
- Changing the characteristics of a job

Aborting and Requeuing Jobs on Output Queues

Aborting and Requeuing Jobs

When you abort a job in a queue, only the aborted job is affected. Any jobs following the aborted job continue to execute.

Use the STOP/QUEUE command to stop a job in an execution queue. This command requires OPER privilege, or ownership of the job.

Format:

STOP/QUEUE [/ABORT] [/REQUEUE] queue-name[:]

- /ABORT causes the current job in the named queue to stop; the next job in line in the queue will then start.
- /REQUEUE stops the current job, then sends it back to the named queue. Use this qualifier when a more important job is waiting for the queue to clear, for example when a large document is printing and a much more important memo must be printed.



Note

The parameters of a queue can be modified by using the SET QUEUE command. However, some situations call for other types of action, such as fixing the printer, adding paper, or replacing the printer ribbon. For these problems it usually is better to stop and restart jobs in a queue.

Example: Requeuing a Print Job

The following example show how to requeue a print job.

```
❶ $ PRINT/QUEUE=MEMOS BIG_UNIMPORTANT.TXT
   Job BIG_UNIMPORTANT (queue MEMOS, entry 547) started on
LPS40$J
❷ $ PRINT/QUEUE=MEMOS SMALL_IMPORTANT.TXT
   Job SMALL_IMPORTANT (queue MEMOS, entry 548) pending
   $ SHOW ENTRY
      Entry      Jobname      Username      Blocks      Status
      -----
❸      547      BIG_UNIMPORTANT SYSTEM      551      Printing
          On busy printer queue LPS40$J

          548      SMALL_IMPORTANT SYSTEM      3      Pending
          On generic printer queue MEMOS
❹ $ STOP/QUEUE/REQUEUE LPS40$J
   $ SHOW ENTRY
      Entry      Jobname      Username      Blocks      Status
      -----
          547      BIG_UNIMPORTANT SYSTEM      551      Pending
          On generic printer queue MEMOS

❺      548      SMALL_IMPORTANT SYSTEM      3      Printing
          On busy printer queue LPS40$J
$
```

Example Notes: Requeuing a Print Job

Notes on the Requeuing a Print Job example include:

- ❶ Send a large, unimportant job to the printer.
- ❷ Send a small, very important job to the printer.
- ❸ The large job may take a very long time to print.
- ❹ Requeue the large job to allow the small job to print first.
- ❺ The small job prints, and the large job is sent back into the queue.

Moving a Job to a Different Queue

It is possible to move a job from one queue to another without re-entering a SUBMIT or PRINT command. One reason to do this is if a job larger than the maximum size specified in the /BLOCK_LIMIT command has been sent to the queue, and the user has logged out.

To move a job from one queue to another, use the SET ENTRY command. This command requires OPER privilege if you are not the owner of the job.

Format:

SET ENTRY entry-number/REQUEUE=queue-name

Example: Moving a Job to a Different Queue

The following example shows the use of the SET ENTRY command to move a print job to a different queue.

```
$ SHOW QUEUE LPS40$JAMED/FULL
Printer queue LPS40$JAMED, idle, on TIDY::JAMED, mounted
form LPS$$FORM
(stock=DEFAULT)
❶ /AUTOSTART_ON=(TIDY::JAMED, *::JAMED) /BASE_PRIORITY=4
/BLOCK_LIMIT=50
/DEFAULT=(FORM=LPS$$FORM (stock=DEFAULT))
/NOENABLE_GENERIC
/LIBRARY=LPS$DEVCTL Lowercase /OWNER=[SYSTEM]
/PROCESSOR=LPS$SMB
/PROTECTION=(S:M,O:D,G:R,W:S) /SCHEDULE=(NOSIZE)
/SEPARATE=(FLAG,
RESET=(LPS$$EOJ), TRAILER)

❷ $ PRINT/QUEUE=LPS40$JAMED REPORT.TXT
Job REPORT (queue LPS40$JAMED, entry 17) pending
pending status caused by job size above queues
maximum job size limit
❸ $ SHOW ENTRY
```

Entry	Jobname	Username	Blocks	Status
17	REPORT	MARSH	137	Pending

On idle printer queue LPS40\$JAMED

```
❹ $ SET ENTRY 17/REQUEUE=LPS40$SCDTST
$ SHOW ENTRY
```

Entry	Jobname	Username	Blocks	Status
17	REPORT	MARSH	137	Printing

On busy printer queue LPS40\$SCDTST

```
$
```

Example Notes: Moving a Job to a Different Queue

Notes on the Moving a Job to a Different Queue example include:

- ❶ Queue LPS40\$JAMED has a maximum block size of 50.
- ❷ Send a large file to the queue.
- ❸ The job will remain in the pending state until the block size limit is changed to allow the job to execute.
- ❹ Move the job to a queue with no block size limit.
- ❺ The job prints on the new queue.

Merging Queues

If you must stop or delete a queue that is not empty, something must be done with the jobs that are in the queue.

Use the ASSIGN/MERGE command to remove all jobs from one queue and merge them into another existing queue.

This command:

- Does not affect jobs that are executing.
- Requires OPER privilege or E (execute) access to both queues.

Format:

ASSIGN/MERGE target-queue source-queue

Example: Merging Queues

The following example shows how to merge queues.

```
❶ $ STOP/QUEUE/NEXT LPB0
❷ $ SET QUEUE/CLOSE LPB0
❸ $ ASSIGN/MERGE LPA0 LPB0
❹ $ DELETE/QUEUE LPB0
  %DELETE-I-DELETED, LPB0 deleted
$
```

Example Notes: Merging Queues

Notes on the Merging Queues example include:

- ❶ Prevent any further jobs from executing on the queue LPB0.
- ❷ Prevent new jobs from being entered into the queue LPB0.
- ❸ Move the jobs from the queue LPB0 to the queue LPA0
- ❹ If appropriate, the queue can now be deleted.

Changing the Characteristics of a Job

Changing Job Characteristics

To change the characteristics of a job in a queue, use the SET ENTRY command. This command requires OPER privilege if you are not the owner of the job.

Format:

SET ENTRY/qualifier entry-number

- The *qualifier* that is used determines the characteristic to be changed.

Example: Changing the Characteristics of a Job

The following example shows the use of the SET ENTRY command to change the characteristics of a print job.

```
❶ SHOW ENTRY/FULL
Entry   Jobname      Username   Blocks    Status
-----
      28   TEST        ROUNDS          1   Holding until 27-
SEP-1999 20:00:00.00
          On generic printer queue ANSI0
          Submitted 27-SEP-1999 09:56:09.24 /FORM=DEFAULT
/PRIORITY=100
          File: _$1$DUA1:[ROUNDS]TEST.TXT;1
❷ $ SET ENTRY/JOB_COUNT=5/AFTER=22:00 28
❸ $ SHOW ENTRY/FULL
Entry   Jobname      Username   Blocks    Status
-----
      28   TEST        ROUNDS          5   Holding until 27-
SEP-1999 22:00:00.00
          On generic printer queue ANSI0
          Submitted 27-SEP-1999 09:56:09.24 /FORM=DEFAULT
/JOB_COUNT=5 /PRIORITY=100
          File: _$1$DUA1:[ROUNDS]TEST.TXT;1
$
```

Example Notes: Changing the Characteristics of a Job

Notes on the Changing the Characteristics of a Job example include:

- ❶ Display information about a print job.
- ❷ Change two characteristics of the job: job count and the time the job will be eligible to be printed.
- ❸ Display information about the job. Note the new job count and time.

OpenVMS Management Tools for Windows NT (OMT)

Printer Management

OpenVMS Management Station makes it easy for you to manage a wide range of printers and print queues across multiple OpenVMS Cluster systems and OpenVMS nodes. In addition, the printer monitoring feature allows you to quickly detect and correct printer problems.

You no longer need to maintain complicated command files to control your printer environment. You can create, delete, and manage a printer and its related queues, as well as perform print job management for those printers, from an easy-to-use Windows interface.

- Some of the tasks you can perform include:
- Monitor one or more printers
- Examine and modify printer attributes
- Delete printers and their associated queues
- Create printers and their associated queues
- Examine and modify queue attributes
- Examine and modify job attributes, requeue jobs, delete jobs

Summary

Concepts

OpenVMS Queues

- There are two classes of queues on an OpenVMS system:
 - Execution — Accepts batch or print jobs for processing
 - Generic — Holds jobs until they are transferred to an execution queue
- Execution and generic queues are categorized by the type of function they perform:
 - Batch
 - Output — Print, terminal, or server
- The queue manager performs the following functions:
 - Coordinates the jobs on the system as they move between generic and execution queues
 - Keeps track of changes made to individual queues
 - Maintains a journal file of queue activity
- Additional queue managers:
 - Allow distribution of queue manager work across multiple nodes
 - The division of queue workload to be selected based on the needs of the OpenVMS Cluster
 - The number of queue managers is selectable
- Each job that is submitted to a queue has a priority associated with it.
- Jobs that are waiting in a queue are executed in order of their priority.

Monitoring Queues and Jobs

- Periodically check the queues on the system for conditions such as:
 - Stalled queues
 - Queues that have stopped unexpectedly
 - Batch jobs that appear to be looping
- Use the DCL command `SHOW QUEUE` to monitor the status of entire queues.
- The `SHOW QUEUE` command takes a number of qualifiers that allow you to display selected subsets of all of the queues on a system.
- Monitor the status of jobs belonging to a particular user, or individual print or batch jobs.

Stopping a Queue

- In some situations the best method of deleting an executing job from a queue is to pause the queue, delete the job, then start the queue.
- To perform an orderly shutdown of a queue, use the `STOP/QUEUE/NEXT` command. This command requires OPER privilege.
- To perform the immediate shutdown of a queue, use the `STOP/QUEUE/RESET` command. This command requires OPER privilege.
- To close a queue, use the `SET QUEUE/CLOSE` command. This command requires OPER privilege.
- To shut down all of the queues on a single node in an orderly manner, use the `STOP/QUEUE/ON_NODE` command. This command requires OPER and SYSNAM privileges.
- To shut down the queues on all nodes in an OpenVMS Cluster, use the `STOP/QUEUE/MANAGER/CLUSTER` command. This command requires OPER and SYSNAM privileges.

Starting Queues

- The QUEUE_MANAGER process must be running before queues can be started.
- To start the QUEUE_MANAGER process, use the START/QUEUE/MANAGER command. This command requires OPER and SYSNAM privileges.
- Additional queue manager processes can be used to off load work from the default queue manager.
- To start additional queue manager processes, use the START/QUEUE/MANAGER/NA command. This command requires OPER and SYSNAM privileges.
- Like the default queue manager, additional queue manager processes must be running before the appropriate queues can be started.
- To start an existing queue, use the START/QUEUE command. This command requires OPER privilege.

Changing the Characteristics of a Queue

- The system manager can determine the characteristics of a queue. These include:
 - CPU time limits for jobs in the queue
 - Minimum and maximum size of jobs accepted by the queue
 - For print queues, whether or not a flag or trailer page is printed for each job
 - The order of execution of jobs in the queue
- To change the characteristics of a queue, use the SET QUEUE command.
- Limit print job sizes to balance the load between printers that differ in performance, but otherwise have similar queue characteristics.

Managing Jobs in a Queue

- Individual jobs in a queue may be stopped.
- When you abort a job in a queue, only the aborted job is affected. Any jobs following the aborted job continue to execute.
- Individual jobs may be moved from one queue to another without re-entering a SUBMIT or PRINT command.
- Characteristics of jobs in a queue may be changed using the SET ENTRY command.

Uses for OpenVMS Management Tools for Windows

- Monitor one or more printers
- Examine and modify printer attributes
- Delete printers and their associated queues
- Create printers and their associated queues
- Examine and modify queue attributes
- Examine and modify job attributes, requeue jobs, delete jobs

Commands

Monitoring Queues and Jobs

SHOW QUEUE

Displays the status of queues.

SHOW ENTRY

Displays the status of individual jobs.

Stopping a Queue

STOP/QUEUE

Pauses a queue.

STOP/QUEUE/NEXT

Shuts down a queue (orderly).

STOP/QUEUE/RESET

Shuts down a queue (immediate).

SET QUEUE/CLOSE

Closes a queue.

SET QUEUE/OPEN

Opens a queue.

STOP/QUEUES/ON_NODE

Shuts down all queues on a single node.

STOP/QUEUE/MANAGER/CLUSTER[/NAME_OF_MANAGER]

Shuts down all queues in an OpenVMS Cluster.

Starting Queues

START/QUEUE/MANAGER

Starts the queue manager.

START/QUEUE/MANAGER/NAME_OF_MANAGER

Starts an additional queue manager.

START/QUEUE

Starts an individual queue.

Changing the Characteristics of a Queue

SET QUEUE

Changes the attributes of a queue.

SET QUEUE/[NO]BLOCK_LIMIT

Limits the size of print jobs an output execution queue will process.

SET QUEUE/SCHEDULE=[NO]SIZE

Determines whether pending jobs in an output queue are scheduled for printing based on the size of the job.

Managing Jobs in a Queue

STOP/QUEUE/ABORT

Aborts a job in a queue.

STOP/QUEUE/REQUEUE

Requeues a job to the same queue.

SET ENTRY/REQUEUE

Requeues a job to a different queue.

SET ENTRY

Changes the attributes of a job.

Performing Backups and Restores

Module 5

About This Module

Introduction

One of the best ways to prevent loss of important data is to make backup copies of the data at regular intervals. A backup copy is a reserve copy of the data that is kept in a safe place, on a magnetic tape or a disk.

The most common system management tasks relating to backup of files are save, restore, and list operations.

Objectives

To perform backups and restores, you should be able to:

- Define backup terms, list things to be considered when planning backups, and construct a BACKUP command.
- Make backups of a disk to tape.
- Use command procedures for backups.
- Restore files from backup copies.
- Use the BACKUP command to list the contents of a save set.

Overview

The files and programs stored in public and private disks on the system represent valuable resources of an organization. There are many ways that these files may be lost, including:

- Natural disasters, such as fire or flood
- Equipment failures
- User error

To protect against loss, copies of all files should be made and stored in a separate location. On an OpenVMS system, the BACKUP utility is used to create these copies.

Using BACKUP, it is possible to make a copy of:

- Individual files
- A directory structure and all of the files in it
- All directory structures and files on a disk volume

Topics

In this section the following topics are discussed:

- Backup terminology
- Planning backups
- BACKUP command

Backup Terminology

Types of Backups

Backups are usually performed for entire disks (or volumes), although individual files or directories can also be backed up.

There are two general types of backups:

- **Full (image) backup** — saves a copy of all of the directory structures and files on a disk.
- **Incremental backup** — saves only those files that have changed since the last full or incremental backup



Note

We are trying to make a slight change in the way we teach backups. In most of the industry, what we call a physical backup is known as an image backup, and what we call an image backup is known as a full backup.

A full backup can be further divided into two types:

- Physical (bit for bit)
- Image (file by file)

In this material, when the term **full backup** is used, we are talking about an image (file by file) backup.

Save Sets

When you back up files to tape in a save operation, the files are stored in units called **save sets**.

A save set is a single file that contains 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.

- The save set includes the files that you save with BACKUP and other information that is used by BACKUP.
- Save sets are written in a format that only BACKUP can understand.
- The save set may be copied, renamed, or deleted using standard DCL commands.
- The contents of a save set may only be listed using the BACKUP utility.
- A save set can also be written to disk if you use the /SAVE_SET qualifier.



Note

By using a special format, the utility is able to save the relationship of a file to a directory structure as well as the contents of the file. For example, it can save the fact that FILE.DAT was located in the directory [USER1] and save FILE.DAT itself.

BACKUP treats all magnetic tape files as save sets. If either the input-specifier or the output-specifier is to be a tape, you do not need to use the /SAVE_SET qualifier.

You cannot use save-set specifications as both the input and output specifiers in a BACKUP command line. Because BACKUP treats all tapes as save sets you cannot use BACKUP to copy files from tape to tape or to copy a save set from disk to tape.

Planning Backups

If there are many disks on the system, making backup copies of all disks will be a time consuming process. Each site needs to establish and publish a schedule for preparing backup copies. It is best to perform backup when the activity on the system is as low as possible. If batch production jobs are run on the system, the backup of the disks used by the production jobs should be coordinated with the running of the batch jobs.

If a disk fails and the files must be restored to another disk, the more incremental tapes that have to be read the longer the restore will take. However, a full backup takes longer to perform than an incremental one. Each site must weigh the costs and design a backup schedule to meet its business needs.

Scheduling Backups

Backup copies of the files on your system should be made on a regular and scheduled basis. A backup strategy will include both full and incremental backups. Some possible full backup schedules include:

- Daily with incremental backup each day at noon
- Once a week with incremental backup between the full backups
- Once per month with incremental backup between the full backups

For small systems, it is useful to use a batch job that backs up the entire system daily, resubmitting itself automatically.

Backup Media

Magnetic tape is the most commonly used medium for storing copies of disk files for archiving and protecting files from loss. It is less expensive than disk media, highly portable, and easy to store.

Open Files During a Backup

It is recommended that BACKUP be used when a minimum number of files are open.

- When a file is open for writing during a backup, the file by default is not copied and saved during the backup procedure.
- You can instruct the Backup utility to save open files by using the qualifier /IGNORE=INTERLOCK on the BACKUP command. By using this qualifier, the contents of the file at the moment of backup are saved.

By using the /IGNORE=INTERLOCK qualifier, however, you may be saving inconsistent data, dependent on the applications that are writing to the open files.

Informing Users About Backups

When you perform any type of backup, notify users using the REPLY/ALL command.

```
$ REPLY/ALL "System Backup about to begin"
Reply received on TIDY from user SYSTEM at _TIDY$RTA2:
13:38:20
System Backup about to begin

3 terminals have been notified on NEAT.
4 terminals have been notified on TIDY.
$
```

BACKUP Command

Parts of a BACKUP Command

To invoke online BACKUP, enter an appropriate BACKUP command at the DCL prompt.

The BACKUP command line has three parts:

BACKUP/qualifiers input/qualifiers output/qualifiers

This part...	Performs this function:
BACKUP/qualifiers	Specifies the kind of backup operation that is to be performed. Command qualifiers modify the default action of a BACKUP command.
input/qualifiers	Specifies the input for a backup operation. The input specification can be a file specification, a BACKUP save set specification, or a device name. Input qualifiers can be used to select files or affect the way a save set is handled in a restore operation.
output/qualifiers	Specifies the output for a backup operation. The output specification can be a file specification, a BACKUP save set specification, or a device name. Output qualifiers determine the way files are restored, or control the way a save set is created.

Input and Output Specifications

The form of the input and output specifier will vary, depending on:

- Whether the operation is a save or a restore
- What devices are being used, such as tape or disk
- Whether a single file or group of files is being copied

Using Wildcards in a BACKUP Command

You can use wildcards in BACKUP command lines.

- For the *input* specification, wildcards are permitted in OpenVMS file specifications and in save set specifications if they are on magnetic tape.
- For the *output* specification, wildcards are permitted for Files-11 volumes.
- Wildcards cannot be used in the *output* specification if it is a save set or a volume created by a BACKUP/PHYSICAL or BACKUP/IMAGE operation.

Making Backups of a Disk to Tape

Overview

Backup copies should exist for every file on your system. The more frequently files are updated, the more frequently they should be backed up.

BACKUP is intended to be used primarily by system managers and operators to protect public media. However, anyone can use BACKUP to make personal backup copies and to transport files between OpenVMS systems.

Topics

In this section the following topics are discussed:

- Preparing disks and tapes for backups
- Making a full backup
- Making an incremental backup

Preparing Disks and Tapes for Backups

Preparing Volumes for Backups

The output volume (disk or tape) for a backup operation must be initialized and mounted. You can do this manually or let the Backup utility do it automatically.

Preparing Tapes for Backups

Disk and magnetic tape devices being used on an OpenVMS system must be initialized. The initialization process formats the disk or tape and writes a label on the volume. All former contents of the device are lost.

Initialization can be performed manually using the INITIALIZE command. When using the BACKUP utility, the initialization can be done by BACKUP.

- During a full backup, the output device is initialized by default.
- When the /REWIND qualifier is used to specify the output tape device, the tape will be rewound and initialized before the save set is created.

How BACKUP Mounts Devices

A device must be mounted for the operating system to be able to access the files on the device. For normal file operation the device is mounted as a file-structured device. If the device is not file-structured, it must be mounted as foreign. The BACKUP utility will mount the device appropriately if it is not currently mounted when requested.

Depending upon the kind of backup being performed and the devices being used, the way the device is mounted will be different.

- Tapes are always mounted foreign.
- Input disks are always mounted as file-structured.
- Output disks are mounted as file-structured except on full backups; then they must be mounted as foreign.

Volume Labels

When you initialize a tape, the tape is given a volume label, which is an identifier of one to six characters included as part of the header information on the tape.

- Backup uses the volume label to ensure that you do not create a save set on the wrong magnetic tape, thus unintentionally overwriting existing data.
- You can use the /LABEL qualifier to specify a volume label of up to six characters.
- If the /LABEL qualifier is not used, the first six characters of the save-set name are used to form the volume label.

Making a Full Backup

Full Backup

A full backup saves all the unopen files on a disk along with their directory structures.

Making a Full Backup

To make a full backup to magnetic tape, use the BACKUP command with the /IMAGE and /RECORD qualifiers.

Format:

**BACKUP/IMAGE/RECORD input_desk: -
output_tape:save_set.type**

- The /IMAGE qualifier directs BACKUP to process an entire volume.
 - The output volume must be mounted foreign using the MOUNT/FOREIGN command. BACKUP will automatically do this if the device is not currently mounted.
 - The /IMAGE restore or copy operation initializes the output device thus destroying the previous contents unless directed otherwise.
- The /RECORD qualifier records the date and time of the backup. BACKUP uses this information when subsequent incremental backups are made, saving only those files that have been created or modified since the most recent BACKUP/RECORD command.

If you plan to perform subsequent incremental backups, include the /RECORD qualifier in your command line.



Note

By default, BACKUP does a file by file transfer. Thus, whatever the output destination, the files will be contiguous on that device. Periodic backup and restore operations improve the I/O performance by defragmenting the device.

The following command saves an entire disk volume to tape on MTA0 and names the save set for the disk.

```
$ BACKUP/IMAGE DBA2: MTA0:DBA2.BCK/SAVE_SET
```

/RECORD Qualifier

The /RECORD qualifier is optional and records the date and time that the files are backed up. Use this qualifier if you plan to perform incremental backups between full backups.

```
$ BACKUP/IMAGE/RECORD DBA2: MTA0:DBA2.BCK
```

When a file has been backed up using the /RECORD qualifier, the date the backup was done can be displayed using the DIRECTORY command with the /FULL qualifier.

```
$ DIRECTORY/FULL LOGIN.COM
```

```
Directory USER8:[USER3]
```

```
LOGIN.COM;24                      File ID:      (9126,1,0)
Size:                2/3           Owner:        [12,200]
Created:             2-MAY-1999 09:32:25.05
Revised:             20-AUG-1999 17:51:52.56 (4)
Expires:             <None specified>
Backup:              3-SEP-1999 17:17:23.60
.
.
.
```

/EXACT_ORDER Qualifier

Depending on the other qualifiers you specify on the BACKUP command line, the /EXACT_ORDER output qualifier enables you to:

- Specify the exact order of tapes and labels that you want to use in a BACKUP operation.
- Preserve the existing volume label on a tape.
- Prevent previous volumes of a multivolume save operation from being overwritten.

You cannot use the /EXACT_ORDER qualifier with:

- A label longer than six characters
- The /IGNORE=LABEL_PROCESSING qualifier
- The /REWIND output qualifier

The following sample command uses the /EXACT_ORDER qualifier to specify the exact order of labels for a BACKUP operation:

```
$ BACKUP/IMAGE/RECORD/VERIFY/NOASSIST
 _From: DBA2:
 _To: MTA0:DBA2.BCK/LABEL=(TAPE1,TAPE2,TAPE3)/EXACT_ORDER
```


Example: Full Backup of Disk to Tape

The following example shows a full backup of \$1\$DUA1: to a tape mounted on MUB0:.

```
❶ $ ALLOCATE MUB0:
    %DCL-I-ALLOC, _NEAT$MUB0: allocated
❷ $ SHOW DEVICE/FILES $1$DUA1

    Files accessed on device $1$DUA1: (NEAT) on 25-AUG-1999
    09:16:44.38

    Process name      PID      File name
❸ USER2              00000000  [000000]INDEXF.SYS;1
    USER2              21C00C5E  [ ]TPU$WORK.TPU$WORK;
    USER2              21C00C5E  [USER2]INIT_FINAL.TPU$JOURNAL;1
    .
    .
    .
❹ $ BACKUP/IMAGE/RECORD $1$DUA1:
MUB0:25AUG.BCK/REWIND/LABEL=25AUG1
    %MOUNT-I-MOUNTED, 25AUG1 mounted on _NEAT$MUB0:
    %BACKUP-I-SOFTWERRS, 5 recoverable media errors occurred writing
MUB0:[ ]25AUG.BCK;
    %BACKUP-I-RESUME, resuming operation on volume 2
❺ %MOUNT-I-OPRQST, Please mount volume 25AU02 in device
_NEAT$MUB0:
    BACKUP requests: Saveset 25AUG.BCK, Volume number 02, write
    ENABLED
    .
    .
    .
$
```

Example Notes: Full Backup of Disk to Tape

Notes on the Full Backup of Disk to Tape example include:

- ❶ Reserve the tape drive for your use.
- ❷ Check to see if there are any open files on the disk.
- ❸ These files are open and will *not* be backed up.

If there had been no open files, it would have been possible to dismount the disk, then mount it private to your process; this ensures that no user will use the disk during the backup.

- ❹ Use /RECORD to prepare for subsequent incremental backups.

A large disk will take more than one tape.

Multivolume Backups

When you save the data on a large disk with BACKUP, the save set may span more than one tape depending upon the density of the tape. There are two ways to handle this situation, you can specify:

- One tape drive in the BACKUP command
- Several tape drives in the BACKUP command

BACKUP fits as much data on the first tape as possible. If one tape drive was specified in the command, BACKUP prompts you to remove the tape in the drive and load a second.

If several tape drives were specified in the command, BACKUP continues writing data to the second tape volume, assuming the tape is loaded, on line, and has the correct label. It unloads the first tape and displays a message that the operation is continuing on the second volume.

In a multivolume save-set operation, BACKUP does not initialize the first volume (unless you use the qualifier /REWIND). BACKUP does initialize subsequent volumes.

Volume Labels

Backup determines the volume labels for subsequent volumes as follows:

- If you did not specify a label on the command line, BACKUP uses the save-set name to create a label for the first volume. For subsequent volumes, BACKUP uses the first four characters of the label of the first volume plus the number of the volume in the sequence. For example, to back up a disk USER with the save-set name USERFILES, the first tape is labeled USERFI, the second USER02, the third USER03.
- If you specified a label on the command line using the /LABEL qualifier, the first tape is initialized with the label specified and the subsequent tapes with the first four characters of the label plus the sequence number.

Making an Incremental Backup

Incremental Backup

Incremental backups save only files that have been created or modified since the last image or incremental backup in which the /RECORD qualifier was used.

Making an Incremental Backup

The syntax for incremental backups is the same as for image backups except that you do not use the /IMAGE qualifier.

To make an incremental backup, use the BACKUP command with the /RECORD and /SINCE qualifiers.

Format:

**BACKUP/RECORD/SINCE=BACKUP disk:[*...] -
tape:save-set.type/LABEL=label**

- The /RECORD qualifier records the date and time of the backup. BACKUP uses this information when subsequent incremental backups are taken, saving only those files that have been created or modified since the most recent BACKUP/RECORD command.

If you plan to subsequently perform any incremental backups, you should include the /RECORD qualifier in your command line.

- The /SINCE=BACKUP qualifier selects files that have been created or modified since the last backup.
- You can specify files to be saved, and you can specify input-specifier qualifiers.

Example: Performing an Incremental Backup

The following example shows an incremental backup of \$1\$DUA1: to a tape mounted on MUB0:.

```
❶ $ ALLOCATE MUB0
   %DCL-I-ALLOC, _NEAT$MUB0: allocated
❷ $ BACKUP/SINCE=RECORD/BACKUP $1$DUA1: -
❸ _$ MUB0:DUA1_I_26AUG.BCK/REWIND/LABEL=I26AUG
   %MOUNT-I-MOUNTED, I26AUG mounted on _NEAT$MUB0:
   .
   .
   .
```

Example Notes: Performing an Incremental Backup

Notes on the Performing an Incremental Backup example include:

- ❶ Allocate the tape for your use.
- ❷ The combination of the /SINCE=BACKUP and /RECORD qualifiers make this an incremental backup.
- ❸ The save set will be written to the tape mounted on MUB0:

The name of the save set is DUA1_I_26AUG.BCK

Use the /REWIND qualifier to initialize and rewind the tape before the backup procedure begins.

Using Command Procedures for Backups

Overview

You can use command procedures to reduce the work involved in performing backups. If no operator intervention is required for a backup, these command procedures can be submitted to run as batch jobs during a time when the system is least busy.

Use command procedures for backups to:

- Make sure that system backups are done in a consistent way
- Minimize the chance of error in specifying backups
- Allow for backups to be performed in batch mode

Topics

In this section the following topics are discussed:

- Nightly image backup
- Nightly incremental backup

Nightly Image Backup

Example: Command Procedure for Nightly Image Backups

The following command procedure performs nightly image backups in batch mode. This procedure is only useful if the backup does not take more than one tape, otherwise someone needs to be present at 2:00 a.m. to change tapes. For this command procedure to run, the system must have a batch queue.

```
❶ $ SUBMIT/AFTER="TOMORROW+2:0" SYS$MANAGER:SYSTEM_BACKUP
$!
$ SET NOON
$ ON ERROR THEN GOTO DONE
$ ON CONTROL_Y THEN GOTO DONE
❷ $ SET PROCESS/PRIVILEGE=(OPER,BYPASS)
$!
❸ $ REPLY/ALL "Full System Backup About to Begin."
$ REPLY/ALL "Open Files Will Not Be Saved"
$!
❹ $ BACKUP/IMAGE/RECORD -
❺     DUA0: -
❻     MUA0:FULL_BACKUP.BCK/REWIND
$!
$ WRITE SYS$OUTPUT "-> Completed backup of DUA0 save set"
$ WRITE SYS$OUTPUT ""
$!
$ DONE:
$   DISMOUNT MUA0:
$   EXIT
```

Example Notes: Command Procedure for Nightly Image Backups

Notes on the Command Procedure for Nightly Image Backups example include:

- ❶ The command procedure resubmits itself to run again tomorrow at 2 a.m.
- ❷ OPER privilege is needed for the REPLY command, and BYPASS is needed for the backup operation.
- ❸ Notify users.
- ❹ The /RECORD qualifier records the current date and time in the file header record of each file that is successfully copied.
- ❺ DUA0: is the disk to be backed up.
- ❻ The tape drive is MUA0:, the save set is named FULL_BACKUP.BCK, and it is to be written starting at the beginning of the tape.



Note

Remind students to be sure that a tape is loaded on the specified device and that it is write-enabled (a common error is to forget to place a write ring on the tape).

When the backup is complete, keep the tape in a safe place and not use it again until another image backup is performed.

Nightly Incremental Backup

Example: Command Procedure for Nightly Incremental Backup

The following command procedure performs an incremental backup on disk DUA0: every week night except Friday.

```
❶ $ SUBMIT/AFTER="TOMORROW+2:0"
SYS$MANAGER:INCREMENTAL_BACKUP
$
$ TODAY = F$CVTIME("TODAY",,"WEEKDAY")
❷ $ IF TODAY .EQS. "Friday" THEN GOTO DONE
$!
$ SET NOON
$ ON ERROR THEN GOTO DONE
$ ON CONTROL_Y THEN GOTO DONE
❸ $ SET PROCESS/PRIVILEGE=(OPER,BYPASS)
$!
$ REPLY/ALL "Backup of DUA0: about to begin."
$ REPLY/ALL "Open files will not be saved"
$!
❹ $ BACKUP/RECORD/SINCE=BACKUP -
❺     DUA0: -
❻     MUA0:INCREMENT.BCK/LABEL=INCREM
$!
$ WRITE SYS$OUTPUT "-> Backup completed"
$!
$ DONE:
$   DISMOUNT MUA0:
$   EXIT
```


Example Notes: Command Procedure for Nightly Incremental Backup

Notes on the Command Procedure for Nightly Incremental Backup example include:

- ❶ The batch job resubmits itself to run again tomorrow.
- ❷ This example assumes that an image (full) backup is performed on Friday nights, so an incremental backup will not be done on Fridays.
- ❸ OPER privilege is needed for the REPLY command, and BYPASS is needed for the backup operation.
- ❹ The /RECORD qualifier records the current date and time in the file header record of each file that is successfully copied.

The /SINCE=BACKUP qualifier selects files that have changed since the date recorded in the BACKUP field of their file header records (date of last backup).

- ❺ DUA0: is the disk to be backed up.
- ❻ The tape drive is MUA0, the save set is named INCREMENT.BCK, and the volume label is INCREM.

Restoring Files from Backup Copies

Overview

It may be necessary to restore files from backup copies.

Use the BACKUP command to restore the following:

- All of the files on a disk (or volume or volume set)
- Files from an incremental backup
- All of the files in a specific directory tree (for example, all of the files in a user's main directory and subdirectories)
- One or more specific files

Topics

In this section the following topics are discussed:

- Using the BACKUP command to restore files
- Restoring all files on a disk
- Restoring files from an incremental backup
- Restoring specific files
- Restoring a directory tree

Using the BACKUP Command to Restore Files

BACKUP Restore Operation

A restore operation returns data from a save set to its original file format on the specified output disk.

BACKUP Command

Use the BACKUP command to restore files.

Format:

BACKUP/qualifiers save-set/qualifiers output

- The *save-set* parameter is the file specification for the save set.
 - When you restore files from a save set stored on a disk, include the /SAVE_SET qualifier with the save set specification.
 - The /SAVE_SET qualifier is not necessary if the save set is stored on a tape.
- The *output* parameter is the output file specification.



Note

The procedure for restoring files depends upon the type of restore operation and whether your most recent backup was an image or incremental backup.

Restoring All Files on a Disk

Restoring an Entire Disk

Use the BACKUP/IMAGE command to restore an entire disk from an image backup.

Procedure for Restoring an Entire Disk

To restore the entire contents of a disk when your most recent backup was an image backup:

1. Mount the output disk (*to*) which you will copy the files using the MOUNT/FOREIGN command.
2. Load the media (tape, disk, or diskette) containing the saved backup copy of your disk.
3. Give the BACKUP command with the /IMAGE qualifier, using the following format:
BACKUP/IMAGE device:save-set output
 - The *save-set* parameter is the file specification for the save set.
 - The *output* parameter is the output file specification.
4. If your backup copy is on more than one tape or diskette, repeat step 2 for each tape or diskette.
5. Dismount the disk onto which you just restored the files, using the /NOUNLOAD qualifier.
6. Remount the restored disk.

Initializing the Output Disk

When you use the /IMAGE qualifier in a restore operation, the disk to which you are restoring the files is initialized.

- You will be unable to access any information that was previously stored on the disk.
- If you do not want to use the /IMAGE qualifier (or if the /IMAGE qualifier was not used during the save operation), then you can still restore the directory structure and all of the files by specifying device:[*...] as the output specifier.



Note

If you do not know the name of the save set, do one of the following:

- If the save set is on a disk, use the DIRECTORY command to determine the name of the save set, for example:

```
$ DIRECTORY BACKUP_DISK: [BACKUPS]
```

```
Directory SYS$SYSDEVICE: [BACKUPS]
```

```
19SEP.BCK;1
```

```
Total of 1 file.
```

The save set is named 19SEP.BCK.

- If the save set is on magnetic tape, load the tape and give the following command, substituting the name of the tape drive you use for MTA0:

```
$ BACKUP/LIST/REWIND MTA0:
```

```
Listing of save set(s)
```

```
Save set:      19SEP.BCK
Written by:    SYSTEM
UIC:          [000001,000004]
Date:         19-SEP-1999 22:03:03.63
.
.
.
```

The save set is named 19SEP.BCK.

Example: Restoring a Disk from a Full Backup

The following sample commands show the procedure for restoring an entire disk from a full backup.

❶\$ MOUNT/FOREIGN \$1\$DUA2:

❷\$ BACKUP/IMAGE MTA1:FULL_BACKUP.BCK/REWIND \$1\$DUA2:

❸\$ DISMOUNT \$1\$DUA2:

Example Notes: Restoring a Disk from a Full Backup

Notes on Restoring a Disk from a Full Backup example include:

- ❶ Logically mount the disk \$1\$DUA2. The files will be restored to this disk.
- ❷ Restore the directory structure and all the files from the save set FULL_BACKUP.BCK to the disk \$1\$DUA2.

The /IMAGE qualifier restores a logical duplicate of the original disk, so that the entire directory structure is restored and the files are placed in the proper directories.

- ❸ Logically dismount the disk.

Restoring Files from an Incremental Backup

Order for Restoring Files

Restore files from an incremental backup in chronological order so the most recent versions of files that have had activity will overwrite previous changes.



Note

Some people have difficulty understanding why it is necessary to restore the most recent incremental backup first, rather than work from oldest to newest.

Assume the following:

- A full backup was made on Sunday.
- An incremental backup is made each night.
- On Friday the disk is restored from the full and incremental backups.

Consider a file that is:

- Created on Monday
- Deleted on Tuesday
- Created again (with the same name) on Wednesday

The file created on Monday will have the same name and version number as the file created on Wednesday. If the incremental backups were to be restored starting with Monday, and proceeding in order through the Thursday incremental, the file would be restored from the Monday incremental, but not the Wednesday incremental, because it would have the same name and version number.

By restoring the incremental backups in **reverse** order, starting with Thursday, the most current version of the file is restored.

Procedure for Restoring Files from an Incremental Backup

Use the following procedure to restore files after one or more incremental backups:

1. Mount the disk, using DCL MOUNT commands *to* which you will copy the files, using the /FOREIGN qualifier which prepares the file structure of the disk for restoring.
2. Load the tape, disk, or diskette that contains the most recent image backup of the disk (or volume or volume set).
3. Give the BACKUP command with the /IMAGE qualifier, using the following format:

BACKUP/IMAGE device:save-set output

4. If your backup copy is on more than one tape or diskette, repeat step 2 for each tape or diskette.
5. Dismount the disk onto which you have just restored the files from the image backup, using the /NOUNLOAD qualifier.
6. Mount the disk that you are restoring as a file-structured volume, using the following format:

MOUNT device: volume_label

7. Dismount the media that contained the image backup, and mount the tape, disk, or diskette that contains the most recent incremental backup of the disk (or volume or volume set).
8. Restore your incremental save sets, beginning with the most recent backup. Use the following format to restore an incremental backup:

BACKUP/INCREMENTAL save-set device

9. Continue restoring the incremental backups in *reverse* chronological order until you have processed all of the incremental backups since the most recent image backup. If the incremental backups are on more than one tape, diskette, or disk, then you must mount each of these successively.

When the oldest incremental backup has been processed, the restore operation is complete.

Example: Restoring a Series of Incremental Backups

The following sample commands illustrate the procedure for restoring a series of incremental backups.

- ❶ \$ MOUNT/FOREIGN \$1\$DUA2:
- ❷ \$ BACKUP/IMAGE/RECORD DBA3:WORK_DISK_BACKUP.BCK/SAVE_SET \$1\$DUA2:
- ❸ \$ DISMOUNT/NOUNLOAD \$1\$DUA2:
- ❹ \$ MOUNT \$1\$DUA2: USER1
- ❺ \$ BACKUP/INCREMENTAL DBA3:WORK_DISK_16_JAN.BCK/SAVE_SET \$1\$DUA2:
- ❻ \$ BACKUP/INCREMENTAL DBA3:WORK_DISK_17_JAN.BCK/SAVE_SET \$1\$DUA2:
- ❼ \$ BACKUP/INCREMENTAL DBA3:WORK_DISK_18_JAN.BCK/SAVE_SET \$1\$DUA2:

Example Notes: Restoring a Series of Incremental Backups

Notes on the Restoring a Series of Incremental Backups example include:

- ❶ Logically mount the disk \$1\$DUA2. The files will be restored to this disk.
- ❷ Restore the directory structure and all the files from the save set WORK_DISK_BACKUP.BCK to the disk \$1\$DUA2. This was an image backup, which must be the first save set you restore when you want to restore incremental backup copies.
- ❸ Logically dismount the disk \$1\$DUA2.
- ❹ Remount the disk \$1\$DUA2, this time as USER1, a file-structured volume.
- ❺ Restore the most recent incremental backup.
- ❻ Restore the next most recent incremental backup.
- ❼ Restore the earliest incremental backup.

Restoring the incremental backups in **reverse** chronological order is the most efficient way to restore files. When you have restored the last incremental backup, the restoration process is complete.



Note

This example uses the following assumptions:

- The save set for the image backup is named WORK_DISK_BACKUP.BCK. This save set was created using the BACKUP/IMAGE/RECORD command.
- The save sets for the incremental backups are named as follows:
 - WORK_DISK_16_JAN.BCK
 - WORK_DISK_17_JAN.BCK
 - WORK_DISK_18_JAN.BCK
- Both the image and the incremental backup copies are on the disk named DBA3, which is already mounted.

-
- The files will be restored to disk \$1\$DUA2.
-

Restoring Specific Files

Use the input qualifier /SELECT to restore specific files from a save set.

Format:

BACKUP save-set-spec/SELECT=file-spec output-spec

Example: Restoring Specific Files

The following example shows a restore of specific files from a saveset:

```
$ BACKUP/LOG MUA0:16JUN.SAV/SELECT=[USER7]KITPREP.COM -  
_ $ $1$DUA1:[USER7]
```

```
%MOUNT-I-MOUNTED, 16JUN mounted on _TIDY$MUA0:  
%BACKUP-S-CREATED, created $1$DUA1:[USER7]KITPREP.COM;3  
$
```

Restoring a Directory Tree

BACKUP does not automatically restore files to the directory from which they were saved.

To restore files to the directory from which they were saved, use the directory wildcard character [*...] in the output specifier.

Example: Restoring a Directory Tree

The following example shows a restore of a Directory tree:

```
$ BACKUP/LOG MUA0:16JUN.SAV/SELECT=[USER3...]  
$1$DUA1:[USER3...]  
  
%MOUNT-I-MOUNTED, 16JUN mounted on _TIDY$MUA0:  
%BACKUP-S-CREATED, created $1$DUA1:[USER3]GREET.COM;1  
%BACKUP-S-CREATED, created $1$DUA1:[USER3]LIBRARY.COM;1  
%BACKUP-S-CREATED, created $1$DUA1:[USER3]LOGIN.COM;25  
%BACKUP-S-CREATED, created $1$DUA1:[USER3]SUBMIT.COM;2  
.  
.  
.
```

Listing the Contents of a Save Set

Overview

Listing the contents of a save set can be useful when you want to select individual files from the save set and restore them to a disk.

Because save sets are written in a format that only BACKUP can interpret, the list operation is the only way to display the contents of a save set.

Topics

In this section the following topics are discussed:

- Using BACKUP to list contents of save sets

Using BACKUP to List Contents of Save Sets

Listing a Save Set

To list the contents of a save set, use the BACKUP command with the /LIST qualifier.

Format:

BACKUP/LIST save-set-spec[/SAVE_SET]

- Include the /SAVE_SET qualifier if the save set is located on disk.



Note

To simplify restore operations, you can:

- List all the files in the save set in a file, using the /LIST=file-spec qualifier.
 - Edit the resulting output file, creating a command procedure that restores only those files that you want to restore.
 - In your command procedure, use the same format for the BACKUP command as when you restore individual files at DCL level.
-

Example: Listing a Save Set

In the following example, the BACKUP command is used to list the contents of a save set.

```
❶ $ BACKUP/LIST MUB0:TEST.BCK/SAVE_SET
    Listing of save set(s)

❷ Save set:          TEST.BCK
   Written by:       USER1
   UIC:              [000100,000001]
   Date:             02-SEP-1999 11:47:53.78
   Command:          BACKUP/RECORD $1$DUA2:[USER1...]*.*
MUB0:TEST.BCK/LOG
❸ Operating system:  OpenVMS Alpha version 7.2
   BACKUP version:   V7.2
   CPU ID register:  0B000006
   Node name:        NEAT::
   Written on:        _NEAT$MUB0:
   Block size:        8192
   Group size:        10
   Buffer count:       74

      [USER1]LOGIN.COM;5                      1  30-AUG-1999
09:23
      .
      .
      .
      Total of 5 files, 46 blocks
      End of save set
      $
```

Example Notes: Listing a Save Set

Notes on the Listing a Save Set example include:

- ❶ Enter the BACKUP/LIST command to display the contents of the save set. The tape does not have to be mounted first.
- ❷ Information about the save set is displayed.
- ❸ This example was generated on an OpenVMS Alpha system.

Summary

Concepts

Backup Concepts

- There are two general type of backup:
 - Full: Copies all of the directory structures and files on a disk
 - Incremental: Copies only those files that have changed since the last full or incremental backup
- A save set is a file that contains the files saved during a backup operation, and also information that is used by BACKUP during a restore operation.
- Backup operations should be scheduled on a regular basis.
- Use the REPLY/ALL command to inform users when a backup is performed.
- Use the BACKUP command to perform an online backup.

Making Backups of a Disk to Tape

- The output volume (disk or tape) for a backup operation must be initialized and mounted.
- You can do this manually or let the Backup utility do it automatically.
- Full backups save all the unopen files on a disk along with their directory structures.
- To make a full backup, use the BACKUP command with the /IMAGE and /RECORD qualifiers.
- Incremental backups save only files that have been created or modified since the last image or incremental backup in which the /RECORD qualifier was used.
- To make an incremental backup, use the BACKUP command with the /RECORD and /SINCE qualifiers.

Using Command Procedures for Backups

- You can use command procedures to reduce the work involved in performing backups.
- Command procedures ensure that system backups are done in a consistent way and minimize the chance of error in specifying backups.
- If no operator intervention is required, these command procedures can be submitted to run as batch jobs during a time when the system is least busy.

Restoring Files from Backup Copies

- A BACKUP restore operation returns data from a save set to its original file format on the specified output disk.
- Use the BACKUP/IMAGE command to restore an entire disk from an image backup.
- Restore files from an incremental backup in reverse chronological order so the most recent versions of files that have had activity will overwrite previous changes.
- Use the BACKUP input qualifier /SELECT to restore specific files from a save set.
- To restore files to the directory from which they were saved, use the directory wildcard character [*...] in the output specifier.

Listing the Contents of a Save Set

- BACKUP must be used to list the contents of a save set.
- A save set listing gives information about the directory structure of the saved files.

Commands

Backup Concepts

REPLY/ALL

Informs users about a backup.

BACKUP

Backs up or restores files, or lists contents of a save set.

Making Backups of a Disk to Tape

BACKUP/IMAGE/RECORD

Performs an image (full) backup.

BACKUP/RECORD/SINCE=BACKUP

Performs an incremental backup.

Restoring Files from Backup Copies

/SAVE_SET

BACKUP qualifier used to specify that an input file is a save set.

BACKUP/IMAGE

Restores an entire disk from a backup.

BACKUP/INCREMENTAL

Restores an incremental backup.

/SELECT

BACKUP qualifier used to specify files that are to be restored from a backup.

Listing the Contents of a Save Set

BACKUP/LIST

Lists the contents of a save set.

Shutting Down and Starting Up an OpenVMS System

Module 6

About This Module

An OpenVMS system usually remains running for a long time. Under certain conditions, however, it may be necessary to shut down the system. You should be able to shut the system down in the appropriate way, and bring it back up when circumstances allow it.

Objectives

To perform system startups and shutdowns, you should be able to:

- Shut down an OpenVMS system.
- Start up an OpenVMS system from the default system device.
- Boot satellite nodes from a boot server.

Shutting Down an OpenVMS System

Overview

Sometimes it is necessary to shut down an OpenVMS system. Situations where this might be necessary include:

- An upgrade or reconfiguration of your system software or hardware
- To perform periodic maintenance
- An interruption of electrical or cooling service to your computer room

The operating system provides two types of shutdown procedures:

- **Orderly shutdown** — performs an organized, step-by-step shutdown

You can perform orderly shutdowns in two ways:

- With the SHUTDOWN.COM command procedure
- With the System Management utility (SYSMAN)

The advantage to using SYSMAN is that you can shut down a group of nodes quickly.

- **Emergency shutdown** — provides a means of halting the system immediately under emergency circumstances
 - The system is functioning, but it cannot be shut down in an orderly fashion
 - The system is not responding ("hung")

The OpenVMS operating system provides a command procedure that shuts down the system at a time you specify and notifies system users when that time approaches. Other shutdown methods are also available. You can use them when the usual procedure is not effective.

The operating system also provides a command procedure to start up the system. You can establish default system devices and starting up your system can be as simple as pressing a button.

This section assumes the OpenVMS software has already been configured to run on your system and that you will be responsible for stopping and starting a system from the default system device.

The details of what takes place during startup and shutdown are not discussed here. This section covers what you need to know to perform these tasks.

Topics

In this section the following topics are discussed:

- Orderly shutdown using SHUTDOWN.COM
- Orderly shutdown using SYSMAN
- BACKUP treats relative version -0 as 0, possibly saving the wrong file version.
- Emergency Shutdown

Orderly Shutdown Using SHUTDOWN.COM

What Happens in an Orderly Shutdown

An orderly shutdown is the preferred method of shutting down an OpenVMS system.

During an orderly shutdown with SHUTDOWN.COM, the operating system attempts to avoid interrupting any critical system operations. It warns users of the approaching shutdown and makes every attempt to preserve all system and user data. For this reason, always use the orderly shutdown procedure if circumstances permit.



Note

It is best to use the console terminal for startups and shutdowns so that you can get a paper copy of any problems that might occur.

Procedure to follow for Orderly Shutdown

To run an orderly shutdown:

- Log in a SYSTEM (or some other suitable privileged account).
- Issue the command: `$ @SYS$SYSTEM:SHUTDOWN`
- Respond to message prompts. The system will ask such things as whether you wish to immediately reboot the system and how much time to wait before starting the shutdown process and whether to perform an automatic reboot.
- Wait until the following message is displayed on the console terminal:
`SYSTEM SHUTDOWN COMPLETE`

The following message is also displayed:

`USE CONSOLE TO HALT THE SYSTEM`

- Halt the system after you see this message.



Note

Not all systems will use the same commands or methods for halting the system. Consult the *Upgrade and Installation Manual* for your computer.

Interrupting the Shutdown

To interrupt a shutdown, press Ctrl/Y.

This action cancels all remaining shutdown functions.

If you interrupt a shutdown, you must manually reverse any functions that have already been executed, such as reinstalling images that have been removed. This is another reason for executing shutdowns from a hardcopy terminal; it gives you a record of what the system has accomplished.

Functions of SHUTDOWN.COM

The site-independent shutdown command procedure `SYSSYSTEM:SHUTDOWN.COM`, performs the following functions, not necessarily in the order listed:

- Displays a notice on terminals that a shutdown is planned
 - Reminds interactive users periodically as shutdown time approaches
- Prevents users from logging in (unless they have OPER privilege)
- Shuts down DECnet software
- Stops batch and print queues
- Executes site-specific DCL commands in `SYSUTDWN.COM` (in `SYS$MANAGER`)
- Stops user processes
- Removes installed images
- Dismounts mounted volumes other than the system disk (and, if requested, spins down the volumes)
- Closes the operator log file
- Stop system processes
- Shuts down secondary processors for multiprocessor systems
- Runs the image `SYSSYSTEM:OPCCRASH.EXE` which:
 - Writes error log entries to disk
 - Causes the system paging mechanism to record on disk any file changes that have occurred in memory
 - Dismounts the system disk
 - Displays a completion message on the console

SHUTDOWN.COM Options

SHUTDOWN.COM offers several shutdown options. The two most commonly used are:

- **REMOVE_NODE**

This option is used for nodes that will not rejoin for an extended time. Active quorum is adjusted.

- **CLUSTER_SHUTDOWN**

This option causes a coordinated shutdown of all nodes. Nodes suspend activity just short of complete shutdown. When all nodes reach this level they all shut down together.

Notification of OpenVMS Cluster Members

Before shutting down a node in an OpenVMS Cluster system, you can define the logical name SHUTDOWN\$INFORM_NODES to be a list of cluster members. When SHUTDOWN.COM executes, all nodes listed in the logical name equivalence will be notified about the shutdown.

Sample commands:

```
$ ASSIGN "NODE1,NODE2,NODE3" SHUTDOWNS$INFORM_NODES
```

```
$ DEFINE SHUTDOWNS$INFORM_NODES "NODE1,NODE2,NODE3"
```



Note

Users on the node that is being shut down are always notified about the shutdown, regardless of whether SHUTDOWN\$INFORM_NODES is defined.

Customizing SHUTDOWN.COM

You can customize SHUTDOWN.COM to meet the needs of your site in the following ways:

- Define logical names to control the operations of SHUTDOWN.COM; for example, to define, specify a list of OpenVMS Cluster nodes to be notified when the system is shut down.
- Modify the site-specific shutdown command procedure, SYSHUTDWN.COM

For more information, refer to the *OpenVMS System Manager's Manual*.

Example: Orderly Shutdown

The following is an example of an orderly shutdown of an OpenVMS system using SHUTDOWN.COM

Welcome to OpenVMS 7.2

Username: SYSTEM

Password:

Welcome to OpenVMS 7.2 on node NEAT

Last interactive login on Monday, 25-SEP-1999 10:29

Last non-interactive login on Monday, 25-SEP-1999 10:44

❶ \$ @SYS\$SYSTEM:SHUTDOWN

SHUTDOWN -- Perform an Orderly System Shutdown
on node NEAT

❷ Do you really want to shutdown node NEAT [NO]? YES

❸ How many minutes until final shutdown [0]: 2

❹ Reason for shutdown [Standalone]: UPGRADE SOFTWARE

❺ Do you want to spin down the disk volumes [NO]? YES

❻ Do you want to invoke the site-specific shutdown procedure [YES]?

❼ Should an automatic system reboot be performed [NO]? YES

❽ When will the system be rebooted [shortly via automatic reboot]: 12:30

❾ Shutdown options (enter as a comma-separated list):

REMOVE_NODE Remaining nodes in the cluster should adjust quorum

CLUSTER_SHUTDOWN Enter cluster is shutting down

REBOOT_CHECK Check existence of basic system files

SAVE_FEEDBACK Save AUTOGEN feedback information from this boot

DISABLE_AUTOSTART Disable autostart queues

Shutdown options [NONE]: save_feedback

%SHUTDOWN-I-SAVEFEED, Saving AUTOGEN's feedback information from this boot...

%SHUTDOWN-I-SAVEOK, Feedback info. Saved in
SYS\$ST\$STEM:AGEN\$FEEDBACK.DAT

%SHUTDOWN-I-OPERATOR, This terminal is now an operator's console
%%%%%%%%%%%%%% OPCOM 25-SEP-1999 10:51:35.28 %%%%%%%%%%%%%%%
Operator_NEAT\$Rta1: has been enabled, username SYSTEM

%%%%%%%%%%%%%% OPCOM 25-SEP-1999 10:51:35.29

Operator status for operator_NEAT\$RTA1:

CENTRAL, PRINTER, TAPES, DISKS, DEVICES, CARDS, NETWORK,
CLUSTER,

LICENSE, OPER1, OPER2, OPER3, OPER4, OPER5, OPER6, OPER7, OPER8,
OPEDR9,

OPER10, OPER11, OPER12

❿ %SHUTDOWN-I-DISLOGINS, Interactive logins will now be disabled

%SET-I-INTSET, login interactive limit -= 0, current interactive value = 1

```

%SHUTDOWN-I-SHUTNET, The DECnet network will now be shut down
%%%%%%%%%%%%% OPCOM 25-SEP-1999 10:51:38.23
%%%%%%%%%%%%%
Message from user DFS$COM_ACP on NEAT
%DFS-I-STOPPING, DFS/COM Stopping
11 SHUTDOWN message on NEAT from user SYSTEM at _NEAT$RTA1:
10:51:39
    NEAT will shut down in 2 minutes; back up 12:30. Please log off node NEAT.
    UPGRADE SOFTWARE

%SHUTDOWN-I-DISAUTOSTART, The autostart queue feature will now be
disabled with
    2 minutes remaining until shutdown
12 1 terminal has been notified on NEAT.
    3 terminals have been notified on TIDY.
    2 terminals have been notified on MESSY.
%SHUTDOWN-I-STOPQUEUES, The queues on this node will now be
stopped
    SHUTDOWN message on NEAT from user SYSTEM at _NEAT$RTA1:
10:52:47
    NEAT will shut down in 1 minute; back up 12:30. Please log off node NEAT.
    UPGRADE SOFTWARE
    1 terminals has been notified on NEAT.
    3 terminals have been notified on TIDY.
    2 terminals have been notified on MESSY.

%SHUTDOWN-I-SITESHUT, the site-specific shutdown procedure will now
be invoked

Now stopping the Vaxshare Print Server...
%SHUTDOWN-I-STOOPUSER, all user processes will now be stopped
%SHUTDOWN-I-STOPAUDIT, the security auditing subsystem will now be
shut down
%SHUTDOWN-I-REMOVE, all installed images will now be removed
%SHUTDOWN-I-DISMOUNT, all volumes will now be dismounted
%DISM-I-INSWPGFIL, 2 swap or page files installed on volume
%DISM-I-MARKEDDMT, _NEAT$DUA0: has been marked for dismount
%SHUTDOWN-I-STOPAUDIT, stopping the audit server
%%%%%%%%%%%%% OPCOM 25-SEP-1999 10:54:08.28
%%%%%%%%%%%%%
Message from user SYSTEM on NEAT
    _NEAT$RTA1:, NEAT shutdown was requested by the operator.
    %%%%%%%%%%%%% OPCOM 25-SEP-1999 10:54:09.06
    %%%%%%%%%%%%%
13 Operator _NEAT$RTA1: has been disabled, username SYSTEM

```

Example Notes: Orderly Shutdown

Notes on the Orderly Shutdown example include:

- ❶ Perform an orderly shutdown.
- ❷ This appears only if you have done a SET HOST.
- ❸ If there are users logged in to the system, give them a few minutes to save their work and log out.
- ❹ Tell users why the system is being shut down.
- ❺ If the system will not be rebooted for some time, you may direct volumes other than the system volumes to be spun down.
- ❻ Answering YES will cause the site-specific shutdown procedure SY\$MANAGER:SYSHUTDOWN.COM to be invoked.
- ❼ Answering YES causes the system to attempt to reboot immediately after the shutdown procedure is completed.
- ❽ The answer to this question will be displayed in the shutdown notice the users receive.
- ❾ One or more options may be selected.
- ❿ Logins are disabled. Only people who have OPER privilege can log in.
- ⓫ Users receive notice that the system is shutting down.
- ⓬ A list of notified terminals is given.
- ⓭ The shutdown process is complete.

Fast Orderly Shutdown

A fast **orderly** shutdown can be performed by specifying zero as the number of minutes to shutdown when executing SHUTDOWN.COM; the time saved is the notification time that normally would occur during the orderly shutdown.

Shutdown Command Files

SYSS\$SYSTEM:SHUTDOWN.COM is provided with the OpenVMS system software. An empty file, SYSS\$MANAGER:SYSHSUTDWN.COM, is also provided. If you want to issue site-specific DCL commands during an orderly shutdown, add them to SYSHUTDWN.COM.

Typically, you issue commands to turn off site-specific devices and remove images installed in the SYSTARTUP_VMS.COM file. You can also send extra messages to users, close files, or do other site-specific jobs.

Emergency Shutdown

Emergency Conditions

If something catastrophic is about to happen to your system, or an orderly shutdown using SHUTDOWN.COM has failed, you may elect to perform an emergency shutdown. The following conditions may exist:

- The system is functioning, but it cannot be shut down in an orderly fashion. DCL is available.
- The system is hung; thus, DCL is not available.

Since ordinarily you would not want to reboot after an emergency shutdown, there is no automatic reboot option for this procedure.

Procedure to Follow for Emergency Shutdown

Use the following procedure to perform an Emergency Shutdown:

1. Log in as SYSTEM or from an account with SYSPRIV privilege.
2. Issue the command: `$ RUN SYS$SYSTEM:OPCCRASH`
3. Wait until the following message is displayed on the console terminal:
`SYSTEM SHUTDOWN COMPLETE`
The following message is also displayed:
`USE CONSOLE TO HALT THE SYSTEM`
4. Halt the system after you see this message.



Note

During an emergency shutdown, OPCCRASHL.EXE processes as few things as possible. The procedure:

- Causes the OpenVMS paging mechanism to record on disk any file changes that have occurred in memory (which ensures that all writable section files are updated to their correct state before the system crashes and all in-memory data is lost)
 - Writes physical memory to the dump file (unless the logical name OPC\$NODUMP is defined)
-

Example: Emergency Shutdown

The following is an example of an emergency shutdown of an OpenVMS system.

```
❶ $ RUN SYS$SYSTEM:OPCCRASH

❷ PRIMARY CPU DATABASE = 80734000

CRASH CPU DATABASE = 80734000

Headers and errorlog buffer dumped...

SPT and GPT dumped...

System space (S0) dumped...

Global pages dumped...

.
.
.
*** MEMORY DUMP COMPLETE ***

SYSTEM SHUTDOWN COMPLETE
```

At this point, use the console to halt your system.

Example Notes: Emergency Shutdown

Notes on the Emergency Shutdown example include:

- ❶ Perform an emergency shutdown.

There are no prompts during an emergency shutdown. It takes only a few seconds from the time [Return] is pressed until the system crashes.

- ❷ The following information appears on the console device.

The system writes out information to the system dump file. This will take a few minutes, depending on the type of system and the number of processes that were running.

Emergency Shutdown Without DCL

If the system is not responding (it is in a "hung" state), use one of the following methods to halt the system:

- Press [Ctrl/P] at the console.



Note

In the extreme case where the system is not responding, you may need to press Ctrl/P at the console or press the Halt key.

Both of these methods should bring up the console prompt (>>>). If the system has a separate console CPU, you must type the H command after the (>>>).

Starting Up and OpenVMS System

Overview

At most sites, the same device usually serves as the system device during each startup. For this reason, each OpenVMS processor supports a method of specifying a default system device.

If the system volume is mounted on the default system device, startup is almost automatic.

The default boot device may be a:

- Local disk
- Disk attached to an HSC unit
- Network adapter
- CD-ROM on a remote system

The boot device used will depend on the operating system and configuration you are using. In most cases, a default device has been specified, and the boot command can be as simple as `BOOT`.

The operating system also provides a command procedure to start up the system. You can establish default system devices and starting up your system can be as simple as pressing a button.

This material assumes the OpenVMS software has already been configured to run on your system and that you will be responsible for stopping and starting a system from the default system device.

The details of what takes place during startup and shutdown are not discussed here. This material covers what you need to know to perform these tasks.

Topics

In this section the following topics are discussed:

- Startup command procedures

Startup Command Procedures

Command Procedures

Two important command procedures are executed when an OpenVMS system is booted:

- `STARTUP.COM` — site-independent startup command procedure
- `SYSTARTUP_VMS.COM` — one of several site-specific startup command procedures; this is a general-purpose procedure to which you add commands to perform operations for setting up your site such as mounting public disks.



Note

At this level of instruction, the goal is not to teach the details or contents of system startup files. The student should only be aware that they exist, and what they do in general.

Some startup command procedures are different on VAX and Alpha systems because the following I/O configuration functions are provided by `SYSGEN` on VAX systems and by `SYSMAN` on Alpha systems:

- Connect devices
 - Load I/O device drivers
-

Functions of STARTUP.COM

SYSS\$SYSTEM:STARTUP.COM executes shortly after the BOOT command is entered at the console prompt. It performs the following functions:

- Defines system logical names used by facilities such as:
 - Symbolic debugger
 - Language Processors
 - Help processor
- Creates processes that control:
 - Error Logging
 - System management server
 - Job controller
 - Operator log
 - Security auditing
- Connects and configures devices
- Installs known images to reduce I/O overhead

Never edit STARTUP.COM, instead place site-specific commands in SYSTARTUP_VMS.COM or other site-specific command procedures.



Note

Point out the difference between DCL and CCL:

DCL

Digital Command Language
Interface to OpenVMS System
\$ prompt
Standard OpenVMS commands type

CCL

Console Command Language
Interface to console code
>>>> prompt
Commands vary by processor

CCL is used primarily for:

- Setting up boot parameters
 - Booting the system
 - Gathering processor information
-

Functions of SYSTARTUP_VMS.COM

SYSS\$MANAGER:SYSTARTUP_VMS.COM executes after STARTUP.COM and is used to:

- Start the DECnet software
- Start system queues
- Mount public (user) disks
- Start the LAT software
- Start up layered products
- Define site-specific system logical names
- Establish the maximum number of interactive logins allowed

Booting Satellite Nodes

Overview

Many sites are configured as NI (network interconnect) or MI (mixed interconnect) clusters. These sites have one or more workstations, or **satellite nodes**, which must be booted from a **boot server**. The satellite nodes do not have the OpenVMS operating system installed on their local disk and must load the operating system from the boot server.

The OpenVMS operating system provides a command procedure that shuts down the system at a time you specify and notifies system users when that time approaches. Other shutdown methods are also available. You can use them when the usual procedure is not effective.

The operating system also provides a command procedure to start up the system. You can establish default system devices and starting up your system can be as simple as pressing a button.

Topics

In this section the following topics are discussed:

- Checking the status of boot service
- Enabling boot service
- Disabling boot service
- SERVICE_ON.COM and SERVICE_OFF.COM



Note

This section assumes the OpenVMS software has already been configured to run on your system and that you will be responsible for stopping and starting a system from the default system device.

The details of what takes place during startup and shutdown are not discussed in this material. This material covers what you need to know to perform these tasks.

Many system managers prefer to use CLUSTER_CONFIG.COM to enable and disable satellite boot service automatically, rather than use the manual methods described in this section. See *the OpenVMS Cluster Systems for OpenVMS* for more information about CLUSTER_CONFIG.COM.

Checking the Status of Boot Service

Status of Boot Service

For a satellite to boot, the boot server must have its boot service enabled. If a satellite is not booting, it is possible that boot service has been disabled.

Use the Network Control Program (NCP) to determine the status of boot service on a node.



Note

Explain that NCP is used to manage the DECnet for OpenVMS Phase IV network, but do not go into detail. The purpose of using NCP here is only to enable or disable boot service, and only the commands will be given.

Example: Using NCP to Check Boot Service Status

The following example shows how to check Boot Service Status using NCP:

❶ Welcome to OpenVMS 7.2

Username: SYSTEM

Password:

Welcome to OpenVMS 7.2 on node NEAT
Last interactive login on Monday, 25-SEP-1999 12:48
Last non-interactive login on Monday, 25-SEP-1999
12:57

❷ \$RUN SYS\$SYSTEM:NCP

❸ NCP>SHOW KNOWN CIRCUITS

Known Circuit Volatile Summary as of 25-SEP-1999 13:17:09

Circuit	State	Loopback Name	Adjacent Routing
Node			
QNA-0 (CLEAN)	on		19.1022

❹ NCP>SHOW CIRCUIT QNA-0 CHARACTERISTICS

Circuit Volatile Characteristics as of 25-SEP-1999
13:17:32

Circuit = QNA-0

State	= on
❺ Service	= disabled
Designated router	= 19.1022 (CLEAN)
Cost	= 4
Maximum routers allowed	= 33

Router priority	= 64
Hello timer	= 15
Type	= Ethernet
Adjacent node	= 19.1022 (CLEAN)
Listen timer	= 90

NCP>EXIT
\$

Example Notes: Using NCP to Check Boot Service Status

Notes on the Using NCP to Check Boot Service Status example include:

- ❶ Log in to the boot server node.
- ❷ Run the network control program (NCP).
- ❸ Get the name of the DECnet circuit, the names are different for different adapters.
- ❹ Use the name of the circuit in the SHOWCIRCUIT command.
- ❺ If the service is disabled, satellites will not be able to boot.



Note

At this point, the only characteristic of interest is whether or not service is enabled.

One way to see if a node will boot a particular satellite is to use NCP to look for rooted directory and loader information. This example shows what is displayed if a satellite is booted from the server:

```
$ RUN SYS$SYSTEM:NCP
NCP>SHOW NODE NEAT CHARACTERISTICS

Node Volatile Characteristics as of 25-SEP-1999
18:-4:28
Remote node = 62.642 (NEAT)

Hardware address           = 08-00-2B-2A-D6-FC
Tertiary loader            =
SYS$SYSTEM:TERTIARY_VMB.EXE
Load Assistant Agent
=SYS$SHARE:NISCS_LAA.EXE
Load Assist Parameter      =
DISK$VAXVMSRL052:<SYS1A.>
```

If the node does not have boot information for the satellite, the following message is displayed:

```
NCP>SHOW NODE TIDY CHARACTERISTICS

Node Volatile Characteristics as of 25-SEP-1999
18:06:53
Remote node = 2.230 (TIDY)

No information available
```

Enabling Boot Service

Boot Service

If boot service is disabled on the boot node, satellites will not be able to boot. The satellite will display a series of RETRY messages as it attempts to boot.

Use the NCP utility to enable satellite boot service manually.



Note

If boot service is disabled, satellites attempting to boot will display RETRY on their console, and the console on the boot node will display something similar to:

```
%%%%%%%%%% OPCOM 25-SEP-1999 18:14:87 %%%%%%%%%%
Message from user DECNET on NEAT
DECnet event 0.7, aborted service request
From node 19.424 (NEAT), 25-SEP-1999 18:14:71
Circuit QNA-0, Line open error, Ethernet address
= 08-00-2B-0A-1C-E6
```

Example: Enabling Satellite Boot Service

The following example shows how to enable satellite boot service using NCP:

```
❶ $ RUN SYS$SYSTEM:NCP
❷ NCP>SET CIRCUIT QNA-0 STATE OFF
❸ NCP>SET CIRCUIT QNA-0 SERVICE ENABLED STATE ON
NCP>SHOW CIRCUIT QNA-0 CHARACTERISTICS

Circuit Volatile Characteristics as of 25-SEP-1999 14:35:08

Circuit = QNA-0

State                      = on
❹ Service                  = enabled
Designated router         = 62.1022 (NEAT)
Cost                      = 4
Maximum routers allowed   = 33
Router priority           = 64
Hello timer               = 15
Type                      = Ethernet
Adjacent node             = 62.1022 (NEAT)
Listen timer              = 90

NCP>EXIT
$
```

Example Notes: Enabling Satellite Boot Service

Notes on the Enabling satellite Boot Service example include:

- ❶ Run the network control program (NCP).

Make sure NCP is run on the boot node.

- ❷ Turn the circuit off so that it can be modified.

The circuit must be in the OFF state before any changes are made to it.

- ❸ Turn the circuit back on and enable boot service.

Do not allow more than a few seconds to elapse between setting the circuit state to OFF and setting it back to ON. If you wait more than RECNXINTERVAL, synchronization will be lost, and you could cause the cluster to lose connection to its satellites.

- ❹ Satellites can now boot.

Disabling Boot Service

Boot Service

When a satellite requests boot service, it sends a boot request onto the network. Each boot server on the network must decide whether it is able to satisfy the request.

In a large network, the number of boot requests that must be evaluated may degrade the performance of the boot node. In such cases, boot service may be turned off on selected boot servers. A disadvantage to this approach is that satellites will not be able to reboot automatically until service is enabled.

Example: Disabling Satellite Boot Service

The following example shows how to disable satellite boot service using NCP:

```
❶ $ RUN SYS$SYSTEM:NCP
❷ NCP>SET CIRCUIT QNA-0 STATE OFF
❸ NCP>SET CIRCUIT QNA-0 SERVICE DISABLED STATE ON
NCP>SHOW CIRCUIT QNA-0 CHARACTERISTICS
```

Circuit Volatile Characteristics as of 25-SEP-1999 154:36:08

Circuit = QNA-0

State	= on
❹ Service	= disabled
Designated router	= 62.1022 (NEAT)
Cost	= 4
Maximum routers allowed	= 33
Router priority	= 64
Hello timer	= 15
Type	= Ethernet
Adjacent node	= 62.1022 (NEAT)
Listen timer	= 90

NCP>EXIT

\$

Example Notes: Disabling Satellite Boot Service

Notes on the Disabling Satellite Boot Service example include:

- ❶ Run the network control program (NCP).
- ❷ Turn the circuit of so that it may be modified.
- ❸ Disable boot service.
- ❹ Satellites will now not be able to boot.



Note

Some overhead is involved in servicing a boot request. Each time a satellite boots, it periodically sends out boot request packets onto the Ethernet. Every machine that has service enabled will look to see if it has boot information for the satellite.

On a small boot server, an excessive number of boot requests can impact performance. This may occur at sites with several clusters on a single Ethernet segment.

One solution is to turn service off at the boot server until it is needed. This may not be necessary at sites with just a few satellites.

SERVICE_ON.COM and SERVICE_OFF.COM

Command Procedures

Two short DCL command procedures are useful for turning satellite boot service on and off. The advantage of these command procedures is that you can enable and disable satellite boot services very quickly, without losing network connections.

Place the command procedures in SYS\$MANAGER on the boot node, and invoke them with the following commands:

```
@SYS$MANAGER:SERVICE_ON
@SYS$MANAGER:SERVICE_OFF
```

Example: SERVICE_ON.COM

The following example shows how SERVICE_ON.COM is used:

```
$! SERVICE_ON.COM
$! Use to enable satellite boot service on a boot node
$!
$ SET NOON
$NCP := $SYS$SYSTEM:NCP
$!
$Replace QNA-0 with the circuit name on your boot server
$!
$ NCP SET CIRCUIT QNA-0 STATE OFF
$ NCP SET CIRCUIT QNA-0 SERVICE ENABLED STATE ON
$
```

Example: SERVICE_OFF.COM

The following example shows how SERVICE_OFF.Com is used:

```
$! SERVICE_OFF.COM
$! Use to disable satellite boot service on a boot node
$!
$ SET NOON
$NCP := $SYS$SYSTEM:NCP
$!
$Replace QNA-0 with the circuit name on your boot server
$!
$ NCP SET CIRCUIT QNA-0 STATE OFF
$ NCP SET CIRCUIT QNA-0 SERVICE DISABLED STATE ON
$
```

Summary

Concepts

Shutting Down an OpenVMS System

- To perform an orderly shutdown of an OpenVMS system, use SHUTDOWN.COM or SYSMAN commands.
- To perform an emergency shutdown of an OpenVMS system, use OPCCRASH.EXE

Starting Up an OpenVMS System

- An OpenVMS system is typically booted from a default system device.
- Once the default device has been specified, use the following startup procedure:
 - Set the appropriate console switches for booting.
 - If the power is off, turn it on.
 - Boot the system using the BOOT command from the console.
- Several command procedures are executed at the startup, including:
 - STARTUP.COM, the general startup procedure
 - SYSTARTUP_VMS.COM, site-specific commands

Booting Satellite Nodes

- For a satellite to boot, the boot server must have its boot service enabled.
- Use the Network Control Program (NCP) to determine the status of boot service on a node.
- Use the Network Control Program to enable or disable satellite boot service on a boot server.
- Two short DCL command procedures, SERVICE_ON.COM and SERVICE_OFF.COM, are useful for turning satellite boot service on and off quickly, without losing network connections.

Commands

Shutting Down an Open VMS System

@SYSS\$SYSTEM:SHUTDOWN

Shuts down an OpenVMS system in an orderly fashion.

SYSMAN>SHUTDOWN NODE

Shuts down a group of nodes quickly.

RUN SYSS\$SYSTEM:OPCCRASH

Shuts down an OpenVMS system in an emergency.

Booting Satellite Nodes

SHOW CIRCUIT x CHARACTERISTICS

NCP command to determine the status of satellite boot service.

SET CIRCUIT x STATE OFF

SET CIRCUIT x STATE ON SERVICE ENABLED

NCP commands to enable satellite boot service.

SET CIRCUIT x STATE OFF

SET CIRCUIT x STATE ON SERVICE DISABLED

NCP commands to disable satellite boot service.

Learning Check

Module 7

Question

1. Which of the following is **not** one of the four major parts of a VAX Computer system?
 - a. Console subsystem
 - b. CPU
 - c. Main memory
 - d. Vector processor
2. To display detailed information about a device on an OpenVMS system, including the owning UIC, use the following command:
 - a. SHOW DEVICE/ALL
 - b. SHOW DEVICE/FULL
 - c. SHOW DEVICE/ID
 - d. SHOW DEVICE/OWNER
3. All OpenVMS devices have a unique device name in the form *ddcu*. What does the *u* represent?
 - a. Device code
 - b. Device type
 - c. Device unit number
 - d. Hardware controller
4. A network in a limited geographical area, such as in a building, is called a:
 - a. Cluster
 - b. DECnet
 - c. LAN
 - d. WAN

5. Computer systems in a network are called:
 - a. Clusters
 - b. Links
 - c. Members
 - d. Nodes

6. Which of the following is **not** a type of process on an OpenVMS system?
 - a. Batch
 - b. Detached
 - c. Network
 - d. Queued

7. To display the names of any subprocesses attached to a process, use the following command:
 - a. SHOW PROCESS/ALL
 - b. SHOW PROCESS/ATTACHED
 - c. SHOW PROCESS/FULL
 - d. SHOW PROCESS/USER

8. The SET PROCESS/NAME command is used to:
 - a. Change the user name of an OpenVMS account
 - b. Change the name of a process
 - c. Change the name of the current file
 - d. Display all processes with a certain name

9. By using the system logical name SYS\$SYSROOT, an OpenVMS system may have:
 - a. More than one CPU
 - b. Multiple user accounts
 - c. Up to 64,000 open files
 - d. Several versions of the operating system

10. To create a batch process, use the following command:
 - a. SUBMIT
 - b. SPAWN
 - c. CREATE
 - d. RUN/DETACHED

11. Match the following OpenVMS system configurations with their descriptions.

a. Single processor		OpenVMS systems that operate as a single domain by running VAXcluster software
b. Multiprocessor		Two OpenVMS systems connected by a special bus that operate as single domain
c. Multiple processor		Any OpenVMS system and its peripherals
d. Vector processor		A special OpenVMS system set up to handle nonscalar processing

12. What are the two **types** of queues on an OpenVMS system?
 - a. Batch and output
 - b. Batch and generic
 - c. Generic and execution
 - d. Generic and output

13. What command is used to start a queue on an OpenVMS system?
 - a. START/PROCESS=QUEUE
 - b. START QUEUE
 - c. START/QUEUE
 - d. START QUEUE MANAGER

14. What command is used to display the status of an output execution queue?
 - a. SHOW QUEUE/DEVICE
 - b. SHOW QUEUE/EXECUTION
 - c. SHOW QUEUE/OUTPUT
 - d. SHOW QUEUE/STATUS
15. Where is information about each user on an OpenVMS system stored?
 - a. Authorized user file
 - b. System user file
 - c. User authorization file
 - d. User information file
16. What utility is used to add or remove a user from the system?
 - a. AUTHORIZE
 - b. SYSADD
 - c. SYSMAN
 - d. USERADD
17. What are two uses of the OpenVMS Management Station Too for Windows NT?
 - a. Manage User Accounts and Printer Queues
 - b. Manage Disk Space and CPU usage
 - c. Monitor memory and manage current processes
 - d. Initiate remote backup and shutdown
18. Which of the following is **not** a method for managing efficient use of disk space? ____
 - a. Establishing quotas
 - b. Purging files
 - c. Restricting user privileges
 - d. Setting directory version limits

19. What command is used to monitor disk space?
 - a. SHOW DEVICE
 - b. SHOW DISK
 - c. SHOW QUOTA
 - d. SHOW SYSTEM

20. What command is used to suspend a process on an OpenVMS system?
 - a. SET PROCESS/HALT
 - b. SET PROCESS/SUSPEND
 - c. STOP/PROCESS
 - d. SUSPEND/PROCESS

21. What command is used to resume a suspended process on an OpenVMS system?
 - a. RESUME/PROCESS
 - b. SET PROCESS/CONTINUE
 - c. SET PROCESS/RESUME
 - d. START/PROCESS

22. What command is used to stop a process on an OpenVMS system?
 - a. DELETE/PROCESS
 - b. SET PROCESS/STOP
 - c. STOP/ID
 - d. STOP/PROCESS

23. What utility is used to manage processes on remote nodes?
 - a. AUTHORIZE
 - b. LATCP
 - c. NCP
 - d. SYSMAN

24. What kind of symbol assignment allows you to replace part of a character string with another character string?
- a. Global
 - b. Overlay
 - c. Local
 - d. Substitution
25. Match each logical name with its description.
- a. SYS\$COMMAND Default input stream
 - b. SYS\$ERROR Default output stream
 - c. SYS\$INPUT Initial input stream
 - d. SYS\$OUTPUT Where error messages are written
26. Match each DCL command with its function.
- a. CLOSE Allows you to access a file
 - b. OPEN Puts data into a file
 - c. READ Takes information from a file
 - d. WRITE Terminates access to a file
27. What startup command procedure initiates error logging, the job controller, and the operator log?
- a. BOOT.COM
 - b. LTLOAD.COM
 - c. STARTNET.COM
 - d. STARTUP.COM
28. In an emergency, what command can be used to shut down an OpenVMS system quickly?
- a. RUN SYS\$MANAGER:OPCRASH
 - b. RUN SYSS\$SYSTEM:EMERGENCY
 - c. RUN SYSS\$SYSTEM:OPCCRASH
 - d. RUN SYSS\$SYSTEM:SHUTDOWN

29. What can you use to determine the status of boot service on a node?
- a. MONITOR
 - b. NCP
 - c. SHOW SYSTEM
 - d. SYSGEN

Learning Check Answers

Module 8

1. Which of the following is **not** one of the four major parts of a VAX Computer system?
 - a. Console subsystem
 - b. CPU
 - c. Main memory
 - d. Vector processor

2. To display detailed information about a device on an OpenVMS system, including the owning UIC, use the following command:
 - a. SHOW DEVICE/ALL
 - b. SHOW DEVICE/FULL
 - c. SHOW DEVICE/ID
 - d. SHOW DEVICE/OWNER

3. All OpenVMS devices have a unique device name in the form *ddcu*. What does the *u* represent?
 - a. Device code
 - b. Device type
 - c. Device unit number
 - d. Hardware controller

4. A network in a limited geographical area, such as in a building, is called a
 - a. Cluster
 - b. DECnet
 - c. LAN
 - d. WAN

5. Computer systems in a network are called Clusters
 - a. Links
 - b. Members
 - c. Nodes

6. Which of the following is **not** a type of process on an OpenVMS system?
 - a. Batch
 - b. Detached
 - c. Network
 - d. Queued

7. To display the names of any subprocesses attached to a process, use the following command:
 - a. SHOW PROCESS/ALL
 - b. SHOW PROCESS/ATTACHED
 - c. SHOW PROCESS/FULL
 - d. SHOW PROCESS/USER

8. The SET PROCESS/NAME command is used to:
 - a. Change the user name of an OpenVMS account
 - b. Change the name of a process
 - c. Change the name of the current file
 - d. Display all processes with a certain name

9. By using the system logical name SYS\$SYSROOT, an OpenVMS system may have:
 - a. More than one CPU
 - b. Multiple user accounts
 - c. Up to 64,000 open files
 - d. Several versions of the operating system

10. To create a batch process, use the following command:
- a. SUBMIT
 - b. SPAWN
 - c. CREATE
 - d. RUN/DETACHED
11. Match the following OpenVMS system configurations with their descriptions.
- | | |
|-----------------------|--|
| a. Single processor | <u>c.</u> OpenVMS systems that operate as a single domain by running VAXcluster software |
| b. Multiprocessor | <u>b.</u> Two OpenVMS systems connected by a special bus that operate as single domain |
| c. Multiple processor | <u>a.</u> Any OpenVMS system and its peripherals |
| d. Vector processor | <u>d.</u> A special OpenVMS system set up to handle nonscalar processing |
12. What are the two **types** of queues on an OpenVMS system?
- a. Batch and output
 - b. Batch and generic
 - c. Generic and execution
 - d. Generic and output
13. What command is used to start a queue on an OpenVMS system?
- a. START/PROCESS=QUEUE
 - b. START QUEUE
 - c. START/QUEUE
 - d. START QUEUE MANAGER

14. What command is used to display the status of an output execution queue?
 - a. SHOW QUEUE/DEVICE
 - b. SHOW QUEUE/EXECUTION
 - c. SHOW QUEUE/OUTPUT
 - d. SHOW QUEUE/STATUS
15. Where is information about each user on an OpenVMS system stored?
 - a. Authorized user file
 - b. System user file
 - c. User authorization file
 - d. User information file
16. What utility is used to add or remove a user from the system?
 - a. AUTHORIZE
 - b. SYSADD
 - c. SYSMAN
 - d. USERADD
17. What are two uses of the OpenVMS Management Station Tool for Windows NT?
 - a. Manage User Accounts and Printer Queues
 - b. Manage Disk Space and CPU usage
 - c. Monitor memory and manage current processes
 - d. Initiate remote backup and shutdown
18. Which of the following is **not** a method for managing efficient use of disk space?
 - a. Establishing quotas
 - b. Purging files
 - c. Restricting user privileges
 - d. Setting directory version limits

19. What command is used to monitor disk space?
 - a. SHOW DEVICE
 - b. SHOW DISK
 - c. SHOW QUOTA
 - d. SHOW SYSTEM

20. What command is used to suspend a process on an OpenVMS system?
 - a. SET PROCESS/HALT
 - b. SET PROCESS/SUSPEND
 - c. STOP/PROCESS
 - d. SUSPEND/PROCESS

21. What command is used to resume a suspended process on an OpenVMS system?
 - a. RESUME/PROCESS
 - b. SET PROCESS/CONTINUE
 - c. SET PROCESS/RESUME
 - d. START/PROCESS

22. What command is used to stop a process on an OpenVMS system?
 - a. DELETE/PROCESS
 - b. SET PROCESS/STOP
 - c. STOP/ID
 - d. STOP/PROCESS

23. What utility is used to manage processes on remote nodes?
 - a. AUTHORIZE
 - b. LATCP
 - c. NCP
 - d. SYSMAN

24. What kind of symbol assignment allows you to replace part of a character string with another character string?
- a. Global
 - b. Overlay
 - c. Local
 - d. Substitution
25. Match each logical name with its description.
- | | |
|-----------------|--|
| a. SYS\$COMMAND | <u>a.</u> Default input stream |
| b. SYS\$ERROR | <u>d.</u> Default output stream |
| c. SYS\$INPUT | <u>c.</u> Initial input stream |
| d. SYS\$OUTPUT | <u>b.</u> Where error messages are written |
26. Match each DCL command with its function.
- | | |
|----------|---|
| a. CLOSE | <u>b.</u> Allows you to access a file |
| b. OPEN | <u>d.</u> Puts data into a file |
| c. READ | <u>c.</u> Takes information from a file |
| d. WRITE | <u>a.</u> Terminates access to a file |
27. What startup command procedure initiates error logging, the job controller, and the operator log?
- a. BOOT.COM
 - b. LTLOAD.COM
 - c. STARTNET.COM
 - d. STARTUP.COM

28. In an emergency, what command can be used to shut down an OpenVMS system quickly?
- a. RUN SYSS\$MANAGER:OPCRASH
 - b. RUN SYSS\$SYSTEM:EMERGENCY
 - c. RUN SYSS\$SYSTEM:OPCCRASH
 - d. RUN SYSS\$SYSTEM:SHUTDOWN
29. What can you use to determine the status of boot service on a node?
- a. MONITOR
 - b. NCP
 - c. SHOW SYSTEM
 - d. SYSGEN

active node

Any VAX processor in a VMSccluster system.

adjacent nodes

Nodes that have a line connecting them.

area

Subgrouping of nodes in a network.

boot server

A node that downline loads the operating system software to satellite nodes.

bridge

A device for connecting two segments of a LAN.

circuit

The logical channel between adjacent nodes.

circuit cost

A positive integer associated with using a circuit.

disk quotas

Disk restriction management based on UICs.

disk server

A VMSccluster node that allows other nodes access to disks to which those nodes are not directly connected.

execution queue

A queue that accepts batch or print jobs for processing.

full (image) backup

Backup operation that saves a copy of all of the directory structures and files on a disk.

gateway

A module, or set of modules, that transforms the conventions of one network into the conventions of another.

generic queue

A queue that holds jobs until they are transferred to an execution queue for processing.

hop

The logical distance to an adjacent node.

incremental backup

Backup operation that saves only those files that have changed since the last full or incremental backup.

LAN

Local area network.

lexical function

Command language construct used to obtain and manipulate information.

line

A physical channel between nodes.

network

An organized system of two or more computers that are linked together either locally or remotely to communicate and share information.

node

A computer system that communicates with other devices in a network.

OMT

OpenVMS Management Station Tools for Windows

OpenVMS Management Station Tools for Windows NT (OMT)

A set of Windows-based tools that allows users to manage their OpenVMS systems from a Windows NT console.

parameter

A value that is passed to a command procedure before execution.

path

A possible route from the source node to the destination node.

path cost

The sum of the circuit costs along a path between two nodes.

path length

The total number of hops along a path between two nodes.

path splitting

The ability of DECnet software to divide the load over multiple paths, if multiple paths exist between a pair of nodes.

process

The working environment on an OpenVMS system.

process-permanent logical name

A logical name that is created for a process by DCL and remains defined for the life of the process.

queue

A list of requests for the system to do something in a particular order.

quota file

One file per disk that contains quota entries.

repeater

Network component used to refresh and regenerate signals.

routing node

A network node that forwards information intended for other nodes.

satellite node

A node that does not have a local system disk, and must load the operating system software from a boot server.

save set

A specially formatted file created by the Backup utility to contain one or more files and their associated directory structures.

server

An entity that is responsible for processing requests.

symbol substitution

Translation of a symbol into its corresponding value.

tape server

A VMScluster node that allows other nodes access to tapes to which those nodes are not directly connected.

volume label

Identifier included as part of the header information on a volume.

WAN

Wide area network.