Educational Services

digital™

VMS System Management I
Student Workbook
Volume I

EY-3505E-SA-0002

July 1989

Contents

About This Course xxii

1 UNDERSTANDING THE USER ENVIRONMENT
1.1 Introduction 1–3
1.2 Objectives 1–3
1.3 Resources 1–4
1.4 Components of the Hardware Environment 1–5
1.4.1 The Central Processing Unit (CPU) 1–5
1.4.2 The Console Subsystem 1–5
1.4.3 Main Memory 1–6
1.4.4 Input/Output Subsystem 1–7
1.5 Interconnect Devices 1–7
1.6 Peripheral Devices 1–8
1.6.1 Terminals 1–8
1.6.2 Printers and Printer/Plotters 1–9
1.6.3 Disk and Tape Drives 1–11
1.6.4 HSC Controllers 1–15
1.7 VMS Device Names 1–16
1.8 Summary 1–17
1.9 System Configurations 1–18
1.9.1 Single Processor Configurations 1–18
1.9.2 Multiple Processor Configurations 1–20
1.9.2.1 Tightly Coupled Configurations 1–22
1.9.2.2 Loosely Coupled Configurations 1–23
1.9.2.3 Clustered Systems 1–27
1.10 Summary 1–35
1.11 Written Exercises 1–37
1.12 Solutions to Written Exercises 1–41
1.13 Review of VMS System Concepts 1–45
1.13.1 The DIGITAL Command Language (DCL) and the VMS Operating System 1–45
1.13.2 Programs, Images, and Utilities 1–46
1.13.3 Processes 1–47
1.13.4 Hardware and Software Contexts 1–48
1.13.5 Virtual Address Space 1–48
1.13.6 Working Sets and Balance Set 1–49
1.13.6.1 Working Set 1–49
1.13.6.2 Balance Set 1–50

1.13.7 Paging 1–50
1.13.7.1 Working Set Extent 1–51
1.13.7.2 Free Page List 1–51
1.13.7.3 Modified Page List 1–52
1.13.8 Scheduling 1–53
1.13.8.1 Priority Levels 1–54
1.13.9 Swapping 1–56
1.13.10 Processor Access Modes 1–57
1.13.10.1 Modes Used Within the VMS Operating System 1–58
1.14 System Files, Directories, and Logical Names 1–61
1.14.1 System Files 1–61
1.14.2 System Directories 1–62
1.14.3 Concealed Root Directories 1–63
1.14.4 System File and Directory Protection 1–64
1.15 Software Available with VMS Systems 1–66
1.15.0.1 Additional References 1–70
1.16 Written Exercises 1–71
1.17 Solutions to Written Exercises 1–73
1.18 User Working Environment 1–74
1.19 Written Exercises 1–81
1.20 Solutions to Written Exercises 1–85
1.21 Learning Aids 1–86
1.21.1 Software Documentation 1–86
1.21.2 Course Offerings 1–86

2 MANAGING SYSTEM USERS
2.1 Introduction 2–3
2.2 Objectives 2–4
2.3 Resources 2–4
2.4 Defining the User Environment 2–5
2.4.1 The User Authorization File (UAF) 2–5
2.4.2 The Authorize Utility 2–9
2.4.3 Creating a User Account 2–17
2.4.3.1 Preparing to Create the Account 2–17
2.4.3.2 Creating the UAF Record and Default Disk Directory 2–19
2.4.3.3 Steps for Adding a User 2–23
2.4.4 Modifying the DEFAULT Record 2–25
2.4.5 Tailoring User Accounts 2–27
2.4.5.1 Identification and Environment Fields 2–28
2.4.6 Login Command Procedures 2–29
2.4.6.1 Access and Security Fields 2–33
2.4.7 Access Times and Modes 2–34
2.4.8 Login Flags 2–36

2.4.9 Security Fields ... 2–38
2.4.9.1 Quotas and Resource Limits ... 2–39
2.4.9.2 Privileges ... 2–41
2.4.10 Removing a User Account ... 2–44
2.4.10.1 Removing the Account ... 2–44
2.4.10.2 Removing the Disk Files ... 2–45
2.4.11 Removing the Disk Quota Record ... 2–45
2.4.12 Removing the VMS Operating System Mail Information ... 2–46
2.4.12.1 Steps for Removing a System User ... 2–47
2.5 Laboratory Exercises ... 2–49
2.6 Solutions to Laboratory Exercises ... 2–50
2.7 Managing Users on a Daily Basis ... 2–51
2.7.1 Restricting User Activity Using DCL Commands ... 2–54
2.8 Laboratory Exercises ... 2–57
2.9 Solutions to Laboratory Exercises ... 2–58
2.9.1 Restricting the Use of Disk Space ... 2–59
2.9.1.1 The SYSMAN Utility and DISKQUOTA Functions ... 2–59
2.9.2 Establishing Quotas on a Volume ... 2–61
2.9.3 Establishing Quotas on a New Volume ... 2–61
2.10 Laboratory Exercises ... 2–71
2.11 Solutions to Laboratory Exercises ... 2–72
2.11.1 Managing Disk Space Using DCL Commands ... 2–74
2.12 Communicating with User Processes ... 2–77
2.12.1 Handling Requests for Operator Assistance ... 2–77
2.12.2 The REPLY Command ... 2–90
2.12.3 The Operator's Log File ... 2–92
2.12.3.1 Additional References ... 2–93
2.13 Laboratory Exercises ... 2–95
2.14 Solutions to Laboratory Exercises ... 2–96

3 MANAGING QUEUES
3.1 Introduction ... 3–3
3.2 Objectives ... 3–4
3.3 Resources ... 3–4
3.4 Overview of Queue Facilities and Operations ... 3–5
3.4.1 The Queue Manager and the System Queue File ... 3–5
3.4.2 Types of Queues ... 3–6
3.4.2.1 Execution Queues ... 3–6
3.4.2.2 Generic Queues ... 3–7
3.4.2.3 Logical Queues ... 3–8
3.5 How the VMS System Handles Print Jobs ... 3–8
3.5.1 Print Job Scheduling ... 3–9

3.6 Print Queue Operations 3–13
3.6.1 Types of Print Queues 3–13
3.6.2 Creating Print Queues 3–13
3.6.2.1 Creating Generic Print Queues 3–17
3.6.2.2 Creating Logical Print Queues 3–18
3.6.2.3 Automatic Queue Creation 3–22
3.7 Laboratory Exercises 3–23
3.8 Solutions to Laboratory Exercises 3–24
3.8.1 Monitoring Print Queues and Jobs 3–25
3.8.1.1 Monitoring Queues 3–25
3.8.1.2 Monitoring Jobs 3–27
3.8.2 Setting Print Queue Attributes 3–29
3.8.2.1 Specifying Separation Pages 3–30
3.8.2.2 Limiting Print Job Sizes 3–37
3.9 Laboratory Exercises 3–39
3.10 Solutions to Laboratory Exercises 3–40
3.10.1 Controlling Print Queues 3–41
3.10.2 Managing Printer Forms and Characteristics 3–46
3.10.2.1 Creating Forms and Characteristics 3–47
3.10.2.2 Using Forms and Characteristics with Printer Queues 3–50
3.10.3 Handling Print Queue Problems 3–51
3.11 Laboratory Exercises 3–53
3.12 Solutions to Laboratory Exercises 3–54
3.13 How the VMS System Handles Batch Jobs 3–55
3.13.1 Batch Job Scheduling 3–58
3.14 Batch Queue Operations 3–60
3.14.1 Types of Batch Queues 3–61
3.14.2 Creating Batch Queues 3–61
3.14.3 Stopping Batch Queues 3–64
3.15 Laboratory Exercises 3–67
3.16 Solutions to Laboratory Exercises 3–68
3.17 Restricting Access and Control of Queues 3–69
3.17.1 UIC-Based Queue Protection 3–69
3.17.2 ACL-Based Queue Protection 3–70
3.18 Overview of Queue Commands 3–72

4 MANAGING DISK AND TAPE VOLUMES
4.1 Introduction 4–3
4.2 Objectives 4–4
4.3 Resources 4–4

4.4 Using Private Disk and Tape Volumes 4–7
4.4.1 Preserving Files 4–7
4.4.2 Transferring Files 4–7
4.4.3 Providing a Private Environment 4–7
4.5 Using Public Volumes 4–8
4.6 Preparing Volumes for Use 4–11
4.6.1 Locating Bad Blocks 4–11
4.6.2 Initializing and Mounting the Volume 4–12
4.6.3 Defining User Access to Volumes 4–13
4.6.3.1 Additional References 4–21
4.7 Obtaining and Modifying Volume Information 4–21
4.8 Using a Public Volume 4–23
4.8.1 Limiting Space Allocation on Public Volumes 4–28
4.8.1.1 Additional References 4–29
4.9 Laboratory Exercises 4–31
4.10 Solutions to Laboratory Exercises 4–32
4.11 Allocating Devices 4–33
4.11.1 Private Volumes 4–33
4.11.2 Public Volumes 4–35
4.11.3 Creating Volume Sets 4–36
4.12 Laboratory Exercises 4–39
4.13 Solutions to Laboratory Exercises 4–40
4.14 Maintaining Private and Public Volumes 4–41
4.14.1 On-Line BACKUP 4–42
4.14.1.1 BACKUP Qualifiers 4–44
4.15 Laboratory Exercises 4–59
4.16 Solutions to Laboratory Exercises 4–60
4.16.1 Operator Action During Multivolume Disk or Tape Set Backups . 4–61
4.16.2 Creating Multivolume Tape Sets on More than One Drive 4–66
4.16.3 BACKUP Tape Label Processing 4–67
4.16.4 BACKUP Journal Files 4–68
4.16.5 Standalone BACKUP 4–71
4.17 Verify Utility 4–71
4.18 Transferring Files Between VAX and PDP-11 Systems 4–74
4.18.0.1 Additional References 4–74

5 CUSTOMIZING THE SYSTEM
5.1 Introduction 5–3
5.2 Objectives 5–4
5.3 Resources 5–4
5.4 The User Environment 5–5

5.5 System Startup Files 5–6
5.5.1 Site-Independent Startup File: STARTUP.COM 5–7
5.5.2 Installing Paging and Swap Files: SYPAGSWPFILES.COM 5–8
5.5.3 Configuring Devices: SYCONFIG.COM 5–8
5.5.4 Defining System-Wide Logical Names: SYLOGICALS.COM 5–9
5.5.5 General Site-Specific Startup Functions: SYSTARTUP_V5.COM. 5–12
5.5.5.1 Mounting Public Disks 5–13
5.5.5.2 Setting Device Characteristics 5–16
5.5.5.3 Initializing and Starting Queues 5–19
5.5.5.4 Installing Known Images 5–19
5.5.5.5 Starting Local DECnet 5–23
5.5.5.6 Starting the LAT Network 5–23
5.5.5.7 Creating Reports About the Last System Failure 5–23
5.5.5.8 Purging Unwanted Versions of Operator Log Files 5–24
5.5.5.9 Setting the Maximum Number of Interactive Logins 5–25
5.5.5.10 Announcing System Availability 5–25
5.6 Laboratory Exercises 5–27
5.7 Solutions to Laboratory Exercises 5–30

6 STARTING UP AND SHUTTING DOWN THE SYSTEM
6.1 Introduction 6–3
6.2 Objectives 6–4
6.3 Resources 6–4
6.4 Starting Up a VMS System 6–5
6.4.1 The VAX Console Subsystem 6–8
6.4.2 The Front Panel Switches 6–9
6.4.3 The Console Processor and Console Device 6–12
6.4.4 The Default System Device 6–15
6.4.5 The Automatic Restart 6–20
6.4.6 Alternate System Devices 6–22
6.4.7 The Standalone Utilities and Diagnostics 6–27
6.4.7.1 Additional References 6–28
6.5 Laboratory Exercises 6–29
6.6 Solutions to Laboratory Exercises 6–31
6.7 Console Commands 6–32
6.7.1 Issuing Console Commands at System Startup 6–32
6.7.2 Issuing Console Commands While the VMS System is Running . 6–34
6.7.3 Customizing Startup and Automatic Restart 6–38
6.7.3.1 Additional References 6–42
6.8 Laboratory Exercises 6–43
6.9 Solutions to Laboratory Exercises 6–44

6.10 System Parameters 6–45
6.10.1 The Conversational Startup 6–45
6.10.1.1 Additional References 6–52
6.11 Laboratory Exercises 6–53
6.12 Solutions to Laboratory Exercises 6–54
6.13 Specifying the System Configuration 6–55
6.14 Multiprocessing Systems 6–57
6.14.1 Starting Up Multiprocessing Systems 6–58
6.14.2 Controlling Multiprocessing 6–58
6.15 Shutdown 6–60
6.15.1 Orderly Shutdown 6–60
6.15.1.1 Automatic Reboot After Normal Shutdown 6–68
6.15.2 The Emergency Shutdown 6–69
6.15.2.1 Automatic Reboot After Emergency Shutdown 6–70
6.15.3 Forcing a Shutdown with CCL Commands 6–70
6.15.3.1 Automatic Reboot After Forcing a Shutdown 6–71
6.16 Laboratory Exercises 6–73
6.17 Solutions to Laboratory Exercises 6–75
6.18 Optional Laboratory Exercise 6–77
6.19 Solutions to Optional Laboratory Exercise 6–79

7 INSTALLING AND UPDATING SYSTEM SOFTWARE
7.1 Introduction 7–3
7.2 Objectives 7–3
7.3 Resources 7–3
7.4 Managing Product Licenses Using the License Management Facility (LMF) 7–5
7.4.1 Overview of the LMF 7–5
7.4.2 LMF Features and Benefits 7–5
7.4.3 Components of the License Management Facility (LMF) 7–7
7.4.3.1 License Units and License Unit Requirement Tables 7–8
7.4.3.2 License Unit Requirement Tables (LURTs) 7–9
7.4.3.3 The License Management Utility (LICENSE) 7–10
7.4.3.4 LICENSE Subcommand Overview 7–12
7.4.4 Messages 7–22
7.5 Software Installation 7–23
7.5.1 Installing a Major Release of the VMS Operating System 7–24
7.5.2 Installing a Major Release 7–24
7.6 Optional Laboratory Exercise 7–27
7.7 Solution to Optional Laboratory Exercise 7–28
7.7.1 Upgrading the VMS Operating System to Version 5.0 7–29
7.8 Optional Laboratory Exercise 7–31
7.9 Solution to Optional Laboratory Exercise 7–32
7.9.1 Installing Maintenance Updates 7–33

7.10	Optional Laboratory Exercise	7–35
7.11	Solution to Optional Laboratory Exercise	7–36
7.11.1	Running UETP	7–37
7.12	Optional Laboratory Exercise	7–39
7.13	Solution to Optional Laboratory Exercise	7–40
7.13.1	Installing Optional (Layered) Products	7–41
7.14	Optional Laboratory Exercise	7–47
7.15	Solution to Optional Laboratory Exercise	7–48
7.16	Customizing and Backing Up the Console Volume	7–49
7.17	Optional Laboratory Exercise	7–53
7.18	Solution to Optional Laboratory Exercise	7–54
7.19	Creating a New System Disk from Another System Disk	7–55
7.20	Standalone BACKUP	7–56
7.20.1	Creating a Standalone BACKUP Kit	7–56
7.20.2	Using a Standalone BACKUP Kit	7–57
7.20.3	Using a Standalone BACKUP Kit	7–61
7.21	Optional Laboratory Exercise	7–65
7.22	Solution to Optional Laboratory Exercise	7–67
8	MAINTAINING SYSTEM INTEGRITY	
8.1	Introduction	8–3
8.2	Objectives	8–3
8.3	Resources	8–4
8.4	Hardware Maintenance	8–5
8.4.1	Handling and Storing Media	8–5
8.4.2	Cleaning Media	8–6
8.4.3	Maintaining the Environment	8–7
8.4.4	Preventive Maintenance	8–7
8.5	Software Maintenance	8–8
8.5.1	System Security	8–9
8.5.2	Physical Security	8–9
8.5.3	Software Security	8–10
8.5.3.1	Erase-on-Delete and Erase-on-Allocate	8–11
8.5.3.2	Login Security	8–13
8.5.3.3	Additional References	8–17
8.5.3.4	UIC and ACL Protection	8–23
8.5.3.5	Additional References	8–39
8.6	Laboratory Exercises	8–41
8.7	Solutions to Laboratory Exercises	8–43
8.7.1	Security Auditing	8–45
8.7.2	Audit Analysis	8–52
8.8	Laboratory Exercises	8–53
8.9	Solutions to Laboratory Exercises	8–55

8.10 System Problems 8–56
8.10.1 Software Problems 8–57
8.10.2 Hardware Problems 8–58
8.10.2.1 The Error Logging Facility 8–58
8.10.2.2 The System Failure Dump Facility 8–63
8.10.2.3 Hardware Diagnostics 8–69
8.10.2.4 User and Operator Comments 8–70
8.10.2.5 Operator Log File 8–71
8.10.2.6 UETP 8–72
8.10.2.7 Additional References 8–72
8.11 Laboratory Exercises 8–73
8.12 Solutions to Laboratory Exercises 8–75
8.13 Optional Laboratory Exercises 8–77
8.14 Solutions to Optional Laboratory Exercises 8–79

9 MONITORING THE SYSTEM
9.1 Introduction 9–3
9.2 Objectives 9–3
9.3 Resources 9–4
9.4 Monitoring System Activity 9–5
9.4.1 Monitoring Active Processes 9–6
9.4.2 Monitoring System Processes 9–11
9.4.3 Obtaining Information About a Device 9–12
9.4.4 Monitoring Memory Resources 9–15
9.4.5 Monitoring Print and Batch Queues 9–17
9.4.6 The Monitor Utility 9–17
9.4.7 Information on a Specific Process 9–24
9.4.8 Interactive Users 9–25
9.4.8.1 Additional References 9–26
9.5 Laboratory Exercises 9–27
9.6 Solutions to Laboratory Exercises 9–29
9.6.1 Collecting Process Information with the Accounting Utility 9–30
9.6.2 Using the Accounting Utility to Produce Reports 9–33
9.7 Laboratory Exercises 9–43
9.8 Solutions to Laboratory Exercises 9–44
9.9 Maintaining System Performance 9–46
9.9.1 Reconfiguring the System with AUTOGEN 9–48
9.9.2 Reconfiguring the System with SYSGEN 9–50
9.9.3 Running AUTOGEN 9–51
9.9.3.1 Modifying System Parameters Without Changing File Sizes 9–51
9.9.3.2 Changing System File Sizes 9–52
9.9.3.3 Additional References 9–55

9.9.4 Performance Tuning 9–56
9.9.4.1 Additional References 9–56
9.10 Laboratory Exercises 9–57
9.11 Solutions to Laboratory Exercises 9–59

Examples

1–1 Displaying the Values of Your Process Parameters 1–78
1–2 Process Parameters of an Interactive Process 1–82
2–1 UAF Record 2–6
2–2 Brief List of UAF Records 2–16
2–3 Adding a System User 2–24
2–4 UAF Record Field Categories 2–28
2–5 Using a Turnkey Account 2–31
2–6 Deleting a UFD and its Subdirectories 2–46
2–7 Removing a System User 2–48
2–8 Volume Quota File Records 2–64
2–9 Using the REQUEST/REPLY Command 2–81
2–10 Operator Aborts a Request 2–82
2–11 Operator Receives Message from REQUEST/REPLY Command 2–83
2–12 Operator Receives Message from MOUNT Program 2–83
2–13 Operator Receives Message from REQUEST Command 2–84
2–14 Operator Receives Two Messages from Authorize Utility 2–84
2–15 Request-Reply Interaction Between User JONES and User TAPES .. 2–88
3–1 JOB_CONTROL and Print Symbiont Processes 3–10
3–2 Scheduling Print Jobs 3–11
3–3 Queue Status Display of Current, Pending, and Holding Jobs 3–21
3–4 Queue Startup Commands in SYSTARTUP_V5.COM 3–22
3–5 Modifying a Running Queue 3–30
3–6 Displaying Queue Forms and Characteristics 3–49
3–7 JOB_CONTROL, Input Symbiont and Batch Job Processes 3–57
3–8 Current and Pending Jobs on a Batch Queue 3–59
3–9 Stopping Batch Queues 3–65
4–1 SHOW DEVICE/FULL Command 4–23
4–2 Initializing and Mounting a Public Disk 4–24
4–3 Creating a Volume Set from an Existing Volume 4–37
4–4 Mounting a Disk with an Unknown Label 4–41
4–5 Listing the Contents of a Save Set 4–52
4–6 Restoring Specific Files from a Save Set 4–55
4–7 Creating a Multivolume Tape Set on One Drive 4–63
4–8 Using the Verify Utility 4–73
5–1 Assigning Site-Specific System Logical Names (SYLOGICALS.COM) . 5–12
5–2 SYSTARTUP_V5.COM Command Procedure 5–15

5–3 Mounting Site-Specific Volumes (MOUNTDSK.COM) 5–16
5–4 Setting Device Characteristics (TERMINALS.COM) 5–19
5–5 Installing Known Images (INSTALL.COM) 5–20
5–6 System Failure Report Procedure (REPORT_FAILURE.COM) 5–24
5–7 Announcing System Availability (START_ANNOUNCE.COM) 5–25
6–1 Default Startup of a VAX-11/780 from Power Down 6–17
6–2 Default Startup of a VAX-11/750 Using the Default System Disk from Power Down 6–17
6–3 Default Startup of a VAX-11/730 from Power Down 6–18
6–4 Startup of a VAX-11/780 Specifying an RM05 as System Device from Power On 6–25
6–5 Startup of a VAX-11/750 Specifying an RM03 as System Device from Power On 6–26
6–6 Startup of a VAX-11/730 Specifying an RL02 as System Device from Power On 6–26
6–7 Using the Console Terminal Interactively After Startup 6–34
6–8 Using the Console Terminal in Console Mode (VAX-11/780) 6–35
6–9 File DB0BOO.CMD from a VAX-11/780 Console Volume 6–39
6–10 Customizing the Console Volume 6–40
6–11 File RESTAR.CMD from a VAX-11/780 Console Volume 6–41
6–12 Typical Conversational Startup 6–50
6–13 Performing an Orderly System Shutdown on a VAX-11/780 System .. 6–62
6–14 User View of Orderly System Shutdown 6–66
7–1 $ LICENSE/LIST/FULL/HISTORY Output 7–17
7–2 Product Authorization Key 7–20
7–3 VMSLICENSE Session - Part I 7–21
7–4 Sample VMSLICENSE Session - Part II 7–22
7–5 Using the VMSINSTAL Command Procedure to Install VAX FORTRAN 7–43
7–6 Saving the Contents of the Console Volume 7–50
7–7 Building a System Disk with VMSKITBLD.COM 7–55
7–8 Creating a Standalone BACKUP Kit on Console Media 7–58
7–9 Backing Up the System Volume to Tape Using a Standalone BACKUP Kit 7–62
8–1 Break-in Suspect Logs in Successfully 8–20
8–2 Break-in Suspect Becomes an Intruder 8–22
8–3 Authorize Utility Automatically Creates Identifiers 8–30
8–4 Displaying Identifiers and Values in the Rights Database 8–34
8–5 Managing the Rights Database 8–36
8–6 Defining an ACL for a File 8–38
8–7 Security Alarm Message on Console Terminal 8–50

8–8 Portion of Device Error and Volume Changes Report Generated by ANALYZE/ERROR 8–61
8–9 Copying the Dump File at Startup and Creating Reports 8–65
8–10 First Page of Report Produced by SDA Command SHOW CRASH ... 8–66
8–11 Second Page of Report Produced by SDA Command SHOW CRASH.. 8–66
8–12 Third Page of Report Produced by SDA Command SHOW CRASH... 8–67
8–13 Fourth Page of Report Produced by SDA Command SHOW SUMMARY/IMAGE 8–68
9–1 Output from the SHOW SYSTEM/FULL Command 9–8
9–2 SHOW DEVICES/MOUNTED Output 9–14
9–3 SHOW DEVICES/FULL Output 9–14
9–4 SHOW MEMORY Output 9–15
9–5 Invoking the Monitor Utility 9–18
9–6 MONITOR Screen Display of the PAGE Class 9–21
9–7 MONITOR PROCESSES/TOPCPU Screen Display 9–22
9–8 MONITOR SYSTEM Screen Display 9–23
9–9 Output from SHOW PROCESS/CONTINUOUS 9–24
9–10 Output from SHOW USERS Command 9–25
9–11 Accounting Record, Full Format 9–31
9–12 Accounting Records, Brief Format 9–35
9–13 Accounting Report, Summary Format 9–36
9–14 Selecting Accounting Files 9–39
9–15 A MODPARAMS.DAT File 9–52

Figures

1–1 VAX Hardware Subsystems 1–5
1–2 Hard-copy and Video Terminals 1–8
1–3 Printers and Printer/Plotter 1–10
1–4 Disks 1–12
1–5 Disk Drives 1–13
1–6 Tape Media 1–14
1–7 Tape Drives 1–15
1–8 MicroVAX II System 1–19
1–9 VAX 8600 System 1–20
1–10 Tightly Coupled System Configuration 1–22
1–11 DECnet Network 1–24
1–12 Configuration with Terminal Servers 1–26
1–13 CI-Only VAXcluster System Configuration 1–29
1–14 Local Area VAXcluster System Configuration 1–31
1–15 Mixed-Interconnect VAXcluster System Configuration 1–32
1–16 The Process of Translating a Program into an Image 1–46
1–17 Components of a VMS Operating System Process 1–47

1–18 Pages in Memory and in Auxiliary Storage 1–53
1–19 Priority Levels in the VMS Operating System 1–55
1–20 Access Mode Hierarchy 1–58
1–21 Layered Design of the VMS Operating System 1–60
1–22 System Directory Tree 1–63
2–1 Adding and Using a UAF Record 2–13
2–2 Creating a User File Directory (UFD) 2–20
2–3 Adding a Quota Record to a Volume Quota File 2–67
3–1 JOB_CONTROL Process Handles All Print Jobs 3–9
3–2 Print Queue 3–11
3–3 Current, Pending, and Holding Jobs 3–20
3–4 File Separation Burst and Flag Pages 3–33
3–5 File Separation Trailer Pages 3–34
3–6 Job Separation Burst and Flag Pages 3–35
3–7 Job Separation Trailer Page 3–36
3–8 JOB_CONTROL Process Handles All Batch Jobs 3–56
3–9 Jobs on a Batch Queue 3–58
4–1 Volume Manipulation Commands 4–9
4–2 Preparing and Using a Disk or Tape Volume 4–10
4–3 User Access to Files on Disk and Tape Volumes 4–14
4–4 Using Output Media for Backups 4–62
6–1 VMS System Startup Phases 6–7
6–2 VAX Console Control Panels 6–11
6–3 Effect of VMS System Startup on System Parameters 6–47
8–1 Auditing VMS System Security 8–48
9–1 The System Accounting File 9–34
9–2 Using the SYSGEN Utility to Modify System Parameters 9–54

Tables

1 Course Conventions xxxi
1–1 Device Codes 1–16
1–2 Comparing Multiple Processor Configurations 1–21
1–3 System Directories 1–61
1–4 Programs and Utilities Distributed with the VMS Operating System . 1–66
1–5 Optional Programs and Utilities Available for the VMS Operating System 1–68
1–6 Parameters that Identify and Control Interactive Processes 1–75
2–1 Fields in a UAF Record Usually Common to Groups of Users 2–7
2–2 Fields in a UAF Record Usually Unique to Each User 2–8
2–3 Standard User Records in the User Authorization File 2–10
2–4 Starting the Authorize Utility 2–11
2–5 Summary of AUTHORIZE Commands 2–12

2–6	Managing the User Authorization File with the Authorize Utility	2–14
2–7	Creating a User's Default Directory or Other UFD	2–21
2–8	Basic Steps to Add a New User to the System	2–22
2–9	AUTHORIZE Qualifiers for Identification and Environment Fields	2–29
2–10	Typical Login Command Procedures (DCL)	2–32
2–11	Login Access Modes	2–35
2–12	AUTHORIZE Qualifiers for Access Fields	2–36
2–13	Login Flag Parameters	2–37
2–14	AUTHORIZE Qualifiers for Security Fields	2–39
2–15	AUTHORIZE Qualifiers for Quota Fields	2–40
2–16	AUTHORIZE Qualifiers for Privilege Fields	2–42
2–17	The VMS Operating System Privileges	2–42
2–18	Regulating the VMS Operating System Processes	2–51
2–19	Controlling Processes	2–55
2–20	DISKQUOTA Commands Within the SYSMAN Utility	2–60
2–21	Establishing Quotas on a New Volume Called DISK$DATA	2–61
2–22	Fields in a Quota File Record	2–63
2–23	Displaying the Contents of a Volume Quota File	2–65
2–24	Managing Individual Records in the Volume Quota File	2–68
2–25	Establishing Quotas on an Existing Volume	2–69
2–26	Controlling Files with DCL Commands	2–75
2–27	Controlling Directories and Volumes with DCL Commands	2–76
2–28	Communication Methods	2–77
2–29	Operator Categories Enabled/Disabled with the REPLY Command	2–78
2–30	Events Requiring Operator Assistance	2–80
2–31	Providing Operator Assistance	2–86
2–32	Sending Messages to Users	2–89
2–33	Qualifiers to the REPLY Command	2–90
2–34	Controlling the Operator's Log	2–92
3–1	Types of Queues	3–6
3–2	Initializing and Starting Queues	3–14
3–3	Establishing Print Queues	3–15
3–4	Creating and Using Print Execution Queues	3–16
3–5	Creating and Using Generic Print Queues	3–18
3–6	Creating and Using Logical Queues	3–19
3–7	SHOW QUEUE Qualifiers for Displaying Types of Queues	3–25
3–8	SHOW QUEUE Qualifiers for Displaying the Amount of Queue Information	3–26
3–9	Queue Status Codes Returned by the SHOW QUEUE Command	3–27
3–10	Job Status Codes	3–28
3–11	Commands to Modify Queue Attributes at Certain Times	3–29
3–12	Job Separation Page Options for the /SEPARATE Qualifier	3–31

3–13 File Separation Page Options for the /DEFAULT Qualifier 3–31
3–14 Setting Block Limits on Print Queues 3–37
3–15 Aborting and Requeuing Jobs 3–42
3–16 Stopping Queues 3–43
3–17 Assigning and Deassigning Logical Queues 3–44
3–18 Moving Jobs from One Queue to Another 3–45
3–19 Deleting a Queue 3–45
3–20 Deleting a Job in a Queue 3–46
3–21 Qualifiers for the DEFINE/FORM Command 3–48
3–22 Defining Printer Forms and Characteristics 3–50
3–23 Positioning a Print Job 3–51
3–24 Aligning Printer Paper 3–52
3–25 Batch Queue Names and Parameter Values 3–62
3–26 Qualifiers to INITIALIZE/QUEUE for Batch Queues 3–63
3–27 User Categories 3–69
3–28 Access to Queues 3–70
3–29 Preparing a Privately Controlled Printer 3–71
3–30 Queue-Related DCL Commands 3–72
4–1 Bad Utility 4–12
4–2 Effects of Access Rights to Files 4–15
4–3 Defining Specific Volume Protection Codes During Initialization 4–17
4–4 Establishing Predefined Volume Protection Codes During Initialization 4–18
4–5 Overriding Volume Protection Codes Established at Initialization 4–19
4–6 Specifying User Access to a Volume 4–20
4–7 Obtaining and Modifying Volume Information 4–21
4–8 Creating and Accessing Private Disk and Tape Volumes 4–27
4–9 Removing Private Disk and Tape Volumes 4–28
4–10 Affecting Space Allocation using INITIALIZE Command Qualifiers 4–29
4–11 BACKUP Terms 4–43
4–12 BACKUP Qualifier Types 4–44
4–13 BACKUP Command Qualifiers 4–45
4–14 BACKUP /IGNORE Qualifier Options 4–47
4–15 BACKUP Input File-Selection Qualifiers 4–48
4–16 BACKUP Input Save Set Qualifiers 4–49
4–17 BACKUP Output File Qualifiers 4–49
4–18 BACKUP Output Save Set Qualifiers 4–50
4–19 Saving Files and Directories with On-Line BACKUP 4–51
4–20 Restoring Save Sets with On-Line BACKUP 4–53
4–21 Backing Up and Restoring Image Volumes 4–57
4–22 Incremental Backup and Restore 4–58
4–23 Generating Labels Automatically 4–67

4–24	Common BACKUP Operations	4–69
4–25	Using the Verify Utility	4–72
4–26	Transferring Files Between VAX and PDP-11 Systems	4–i
5–1	Files Controlling the VMS System Environment	5–6
5–2	Some Standard Logical Names to Define in Site-Specific Startup File	5–10
5–3	Assigning System Logical Names	5–10
5–4	Setting Permanent Characteristics of Terminals	5–17
5–5	Establishing Ownership and Protection of Terminals and Other Non-Shareable Devices	5–18
5–6	Functions of the Install Utility	5–21
5–7	INSTALL Command Qualifiers	5–22
6–1	VAX Console Subsystem	6–8
6–2	VAX System Front Panel Switches	6–9
6–3	VAX System Front Panel Lights	6–10
6–4	VAX Console Devices	6–14
6–5	Starting Up a VMS System from Power Off Using the Default System Device	6–16
6–6	Starting Up a VMS System from Power On Using the Default System Device	6–19
6–7	Automatic Powerfail Recovery	6–21
6–8	Starting Up a VMS System, Explicitly Specifying the System Device	6–23
6–9	Device Codes Used at System Startup	6–24
6–10	VAX Processor and CCL Commands	6–27
6–11	Typical Console Commands	6–33
6–12	Issuing Console Commands While the VMS System is Running	6–37
6–13	Disabling Console Mode	6–37
6–14	CCL Command Files Used at Startup	6–38
6–15	Using SYSBOOT During Conversational Startup	6–46
6–16	Starting Up a VMS System Conversationally	6–49
6–17	Customizing the System Configuration	6–56
6–18	SYSGEN Parameters for Multiprocessing Systems	6–58
6–19	DCL Commands to Control Multiprocessing Systems	6–59
6–20	Forcing a Shutdown Using CCL Commands	6–71
7–1	Six License Unit Requirement Tables (LURTs)	7–8
7–2	Values for an Activity License LURT	7–10
7–3	LICENSE Subcommands	7–12
7–4	Booting the Console Volume	7–60
8–1	Setting Erase-on-Delete for a File or Volume	8–12
8–2	Defining User Passwords	8–13
8–3	Defining a System Password for a Terminal	8–14
8–4	Using Passwords	8–14

8–5 Establishing Ownership and Protection of Terminals and Other Nonshareable Devices 8–16
8–6 SYSGEN Parameters for Break-In Detection 8–19
8–7 Dividing Users into Groups 8–24
8–8 Interaction Between Processes in Same Group 8–25
8–9 ACL- and UIC-Based Protection 8–27
8–10 Terminology Used to Discuss VMS System Access Control 8–28
8–11 Managing the Rights Database Using the AUTHORIZE Command .. 8–32
8–12 Using the SET RIGHTS_LIST Command 8–33
8–13 Auditing Event Classes 8–46
8–14 Defining and Listing Audit Classes 8–51
8–15 Enabling and Disabling Error Logging 8–59
8–16 Selecting Entries for an Error Log Report 8–60
8–17 Analyzing a System Dump 8–64
8–18 Reports Generated by the System Dump Analyzer 8–64
9–1 System, Process, and Device Monitoring 9–5
9–2 System States 9–9
9–3 System Processes 9–11
9–4 Results of System Process Deletion 9–12
9–5 SHOW DEVICES Command 9–13
9–6 Effect of Memory Sizes on Performance 9–16
9–7 MONITOR Class Names 9–19
9–8 MONITOR PROCESSES Class Qualifiers 9–22
9–9 Recording Accounting Information 9–32
9–10 Some Qualifiers Used to Specify Content of Accounting Report 9–37
9–11 Qualifiers Affecting Output Format of Accounting Report 9–38
9–12 Creating an Accounting Report 9–40
9–13 System Files 9–47
9–14 AUTOGEN Phases 9–49
9–15 Using the SYSGEN Utility 9–53

About This Course

Introduction

The *VMS System Management I* course is designed to teach students how to manage a computer running the VMS operating system. It is written in a text-based format, which means you can proceed at a rate comfortable for you. It also provides you ample opportunities for practice and self-testing. Note, however, that the course is not computer-based; there is no computer program that instructs you. Instead, your instruction is provided in the course modules. Certain examples and exercises are provided on-line.

Instead of a teacher, you have a course administrator and a subject matter expert. In some cases, the same person serves as both. The course administrator manages the course in general. The administrator makes sure that you have easy access to the system and on-line course materials. As you finish each module, the administrator records your progress.

The laboratory exercises in this course refer only to the course administrator, but if your site has designated someone as a subject matter expert, you should ask that person about any technical questions you have. Before you consult the expert, however, you are expected to read the course materials and references in an effort to answer the question yourself.

This **Student Workbook** is divided into a number of chapters, or modules, each designed to cover a well-organized topic, or group of topics. Most modules include figures, tables, and examples to enable students to better understand the material. Written and laboratory exercises can be found in each module to allow students to test their VMS system management skills.

This **About This Course** module describes the contents of the course and suggests ways to use its materials most effectively. The following topics are discussed here:

- Resources
- Course Description
- Prerequisites
- Course Organization
- Course Goals
- Nongoals
- Course Conventions
- Course Map

Resources

Students should have access to the following manuals to derive the greatest benefit from this course. Students may be given their own copy of some of these manuals, and your course administrator may provide others for reference during the week.

1. *DIGITAL Terminals and Printers Handbook*, EB-18251-20
2. *VMS DCL Dictionary*, AA-LA12A-TE
3. *VMS DCL Concepts Manual*, AA, LA10A-TE
4. *Guide to Setting Up a VMS System*, AA-LA25A-TE
5. *VMS SYSMAN Utility Manual*, AA-LA26A-TE
6. *VMS Install Utility Manual*, AA-LA29A-TE
7. *Guide to Maintaining a VMS System*, AA-LA34A-TE
8. *VMS Analyze/Disk_Structure Utility Manual*, AA-LA39A-TE
9. *VMS Backup Utility Manual*, AA-LA35A-TE
10. *VMS Mount Utility Manual*, AA-LA38A-TE
11. *VMS AUTHORIZE Utility Manual*, AA-LA42A-TE
12. *VMS Accounting Utility Manual*, AA-LA44A-TE
13. *VMS Monitor Utility Manual*, AA-LA45A-TE
14. *VMS License Management Utility Manual*, AA-LA33A-TE
15. *VMS Software Information Management Handbook*, EB-31479-76
16. *VMS Software Languages and Tools Handbook*, EB-29813-48
17. *VMS System Manager's Manual*, AA-LA00A-TE
18. *VAX Systems and Options Catalog*, EC-I0116-31
19. *VMS System Software Handbook*, EB-30905-48
20. *VMS Installation and Operations Guide* for your system
21. *VMS Software Product Description (SPD)* most recent version

At least one copy of the entire extended VMS operating system documentation set should be available for reference.

Course Description

VMS System Management I is designed to train the system manager or advanced system operator of a VAX computer in running the VMS operating system.

The course gives a theoretical, as well as a practical, insight into system management. In addition to routine system management skills, it introduces tools for monitoring system performance and integrity. It discusses various techniques needed to assist users on the system, back up and restore system and user files, start up and shut down the system, manage system devices, and maintain data security.

Prerequisites

Before taking this course, students should be able to:

- Log in to a VMS operating system
- Use the appropriate VMS system utilities and commands to:
  - — Display on-line help text
  - — Create and manage directories and files
  - — Submit print and batch jobs
  - — Define and use logical names and command synonyms
  - — Display information about the system
  - — Create and maintain private disk volumes
  - — Communicate with other system users

- Develop DCL command procedures that:
  - Control input/output (I/O)
  - Create and access sequential files
  - Use symbols to manipulate constants and variables
  - Use lexical functions
- Define virtual memory and virtual addressing, and describe how virtual memory and physical memory are related
- Define a process and describe process concepts, including:
  - Working set
  - Paging
  - Types of processes
  - States
  - Priority
  - Privileges

These prerequisites can be satisfied by taking the following courses:

- *VMS Utilities and Commands*
- *VMS System Architecture*

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. Each module consists of:

- An **introduction**, which describes the purpose of the module, 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 your study efforts on a selected number of skills.
- The module **text**, which consists of:
  - — Descriptive text organized in a list format
  - — Illustrations, which clarify the relationships among various elements of a VMS system, or summarize steps of a particular process or command
  - — Examples containing sample listings from actual interactive sessions on a VMS system
- A module **summary**, which reviews important concepts and skills taught in the module.

Written and laboratory exercises are also provided with this course. Exercises help students to review and practice the skills learned during the lecture session.

Course Goals

After completing this course, students should be able to:

- Manage system users, which requires:
  - — Maintaining such files as the User Authorization File and volume quota files
  - — Creating user file directories (UFDs)
  - — Controlling user processes
- Manage system resources, which requires:
  - — Managing disk and tape volumes
  - — Defining device characteristics
  - — Creating and managing print and batch queues
  - — Backing up and restoring files and volumes
- Start up and shut down the system
- Customize the system
- Install maintenance updates and optional software
- Establish security measures on the system and audit security-related actions done by users
- Monitor the system for behavior and performance problems, and submit Software Performance Reports when appropriate

Nongoals

- Programming using system services, run-time library routines, or other VMS features (taught in *Utilizing VMS Features* courses)
- Advanced VMS system concepts or system programming (taught in *VMS Internals* courses)
- Details of layered product features and functions (taught in layered product courses)
- Installation under unusual circumstances
- Details of system performance management and tuning (taught in *VMS System Performance Management*)
- Details of system security features (taught in *VMS System Security Features*)
- VAXcluster management (taught in *VAXcluster System Management*)
- Network management (taught in *DECnet Network Management*)
- Creating new DCL commands, help files, and error messages (mentioned in this course, but covered in detail in *VMS System Management II*)
- Troubleshooting error conditions (taught in *VMS System Management II*)
- System configuration (taught in *VMS System Management II*)
- The Files-11 structure of disk and tape volumes (discussed in *VMS System Management II*)
- Converting software or data to the VMS system from another operating system

Course Conventions

Table 1 describes the conventions used in the listings and command tables of this **Student Workbook**.

Table 1 Course Conventions

Convention	Meaning
CTRL/X	Press and hold the key labeled CTRL while you press another key (X). Many control keys have special meanings.

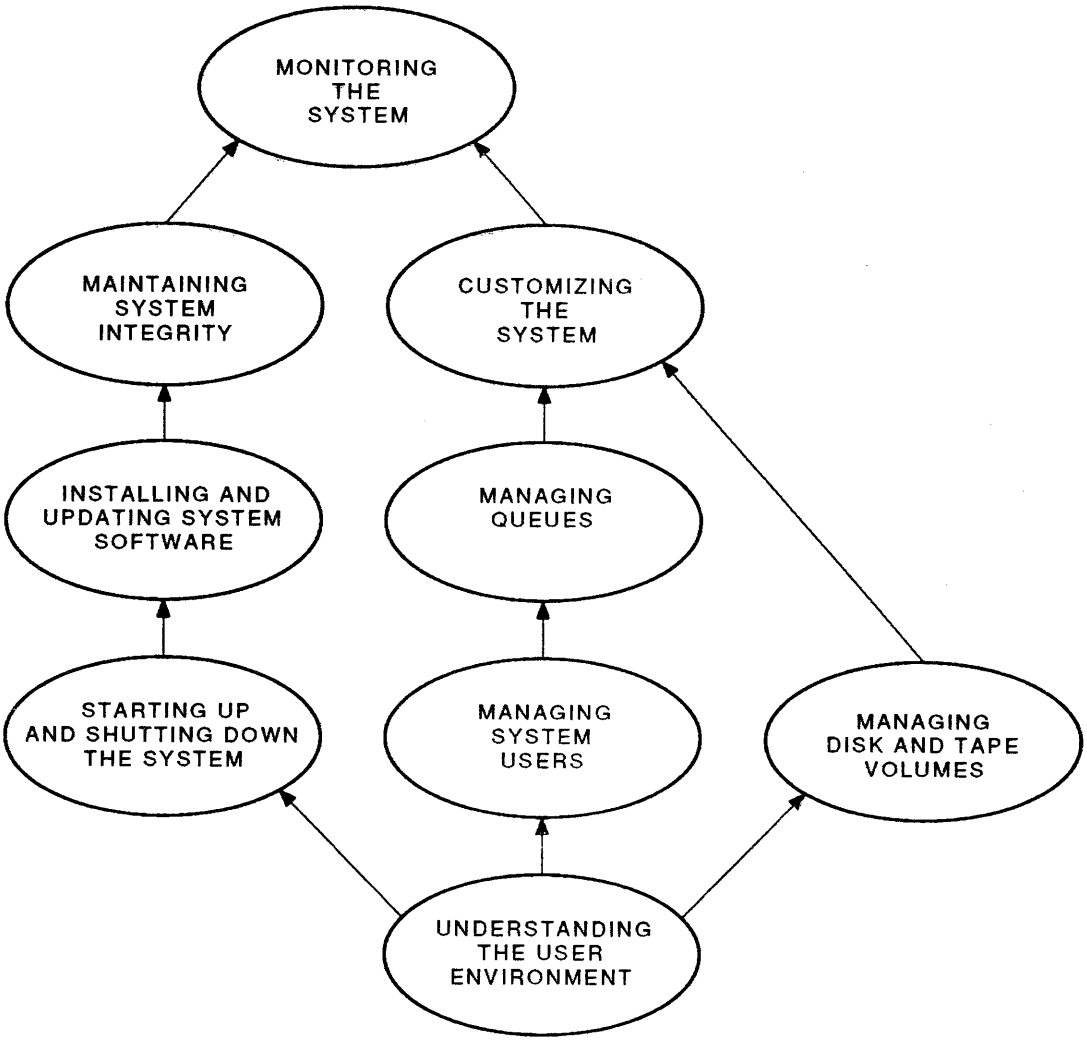
UPPERCASE	In commands, uppercase characters indicate words you type exactly as they appear. For example, type the following commands as they appear:
	
	`$ DIRECTORY`
	`$ TYPE LOGIN.COM`

lowercase	Lowercase characters represent elements that you must replace according to the description in the text. For example, you must follow certain rules when you replace "file-spec" in the following example: `$ TYPE file-spec`

Ellipsis (. . .)	Horizontal ellipses indicate that you can enter additional parameters, values, or information. You can enter any number of file specifications in the following example:
	
	`$ TYPE file-spec, . . .`
	
	Vertical series of periods or ellipses mean that not all data that the system displays in response to the particular command is shown, or that not all data a user enters is shown.
	
	`$ TYPE MYFILE.DAT`
	`.`
	`.`
	`. $`

Square Brackets ([ ])	Square brackets indicate that the enclosed item is optional. (Square brackets are not optional, however, in the syntax of some file specifications.) For example, the logical name is optional in the following command: `$ MOUNT/FOREIGN $TAPE1` Brackets indicate that you must select from the included items.
Quotation Marks (") and Apostrophes (')	The term **quotation** marks refers to double quotation marks. The term **apostrophe** refers to a single quotation mark.

Course Map


MONITORING THE SYSTEM
MAINTAINING SYSTEM INTEGRITY
CUSTOMIZING THE SYSTEM
INSTALLING AND UPDATING SYSTEM SOFTWARE
MANAGING QUEUES
STARTING UP AND SHUTTING DOWN THE SYSTEM
MANAGING SYSTEM USERS
MANAGING DISK AND TAPE VOLUMES
UNDERSTANDING THE USER ENVIRONMENT
TTB_X0847_88

UNDERSTANDING THE USER ENVIRONMENT

1.1 Introduction

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

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

1.2 Objectives

To maintain a VMS system, you should be able to:

- Identify the functions of each component of the hardware environment, namely:
  - — The central processing unit (CPU)
  - — The console subsystem
  - — Main memory
  - — The input/output (I/O) subsystem
- Identify the purpose of an interconnect device, and list some common interconnect devices for VMS systems.
- Recognize the peripheral devices supported by VMS systems.
- Recognize the format for device names on a VMS system.
- List the characteristics of the following types of system configurations:
  - — Single processors
  - — Tightly coupled systems (multiprocessors)
  - — Loosely coupled systems (networks)
  - — VAXcluster systems

- Identify the major software components of the VMS operating system.
- Identify software tools and utilities distributed with the VMS system.
- Identify some optional software products that can be used on VMS systems.
- Identify important system directories, files, and logical names.
  - — Describe the concept and uses of concealed root directories
- Identify the components of a user's process.

1.3 Resources

1. *VMS Installation and Operation Guide* for your particular VAX system
2. *VAX Systems and Options Catalog*
3. *VMS DCL Dictionary*
4. DIGITAL *Terminals and Printers Handbook*
5. DIGITAL *Networks and Communications Buyer's Guide*
6. *User's Guide* for each device in which you are interested
7. *VMS System Software Handbook*
8. *VMS Software Product Description* (SPD) most recent version
9. *VMS Software Information Management Handbook*
10. *VMS Software Languages and Tools Handbook*
11. *VMS DCL Concepts Manual*

1.4 Components of the Hardware Environment

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

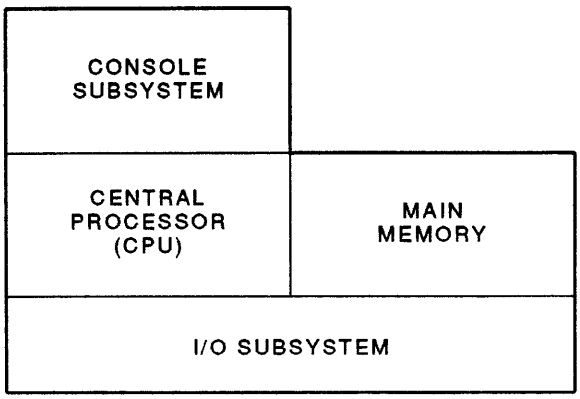


Figure 1–1 VAX Hardware Subsystems

1.4.1 The Central Processing Unit (CPU)

The primary function of the CPU is to execute instructions. It can only execute one instruction at a time. There are several different VAX CPUs, including the MicroVAX II processor, the VAX-11/780 processor, and the VAX 8200 processor. The relative speed at which instructions are processed varies among VAX processors.

1.4.2 The Console Subsystem

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

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

There are several components of the console subsystem, including:

- **Indicators and controls** are located on the processor control panel. These lights and switches are used to monitor and control the system. The processor control panel is different for each VAX processor.

  Some VAX processors do not have a physical control panel with lights and switches. On these processors, the control panel functions are implemented in the console software. The console software allows you to enter commands to perform the control panel functions.

- **The console terminal** is used for starting up and shutting down the system, and for diagnosing problems. It is often a hard-copy terminal, but a video terminal can also be used.

- **The console storage device** is either a diskette drive, TU58 tape cartridge drive, or a hard disk drive. Some VAX systems, such as the MicroVAX II, do not have a dedicated console storage device.

The console subsystem runs in two modes: **console mode** and **program mode**. When the console subsystem is in console mode, use the console command language (CCL) to enter console commands. When the console subsystem is in program mode, use the console terminal like any other terminal on your system.

Using the console subsystem to start up and shut down the system is explained later in this course. You can also refer to the *VMS Installation and Operation Guide* for your particular VAX system for more information on specific console subsystems.

1.4.3 Main Memory

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

- **READ-ONLY memory** (ROM) stores part of the startup sequence for the VAX system. The contents of ROM do not usually change.

- **READ/WRITE memory** stores data and instructions for programs that the CPU executes. The contents of READ/WRITE memory change as different programs are executed.

Each VAX system supports a different amount of memory. System managers can expand the amount of memory up to the maximum amount supported by the particular VAX system. Refer to the *VAX Systems and Options Catalog* for more information on ways to expand the memory on a system.

1.4.4 Input/Output Subsystem

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

- Terminals
- Printers
- Disk drives
- Tape drives

Specific devices are discussed in greater detail in later sections of this module.

1.5 Interconnect Devices

Interconnect devices (also referred to as **buses**) are cables that connect the various subsystems of the computer. The most common function of a bus is to connect peripherals and memory to the processor. There are several buses available for VAX computers, including:

- **UNIBUS** equipment serves as the interconnect for a variety of devices, including terminals and printers.
- **Q-bus** is an interconnect for smaller systems. The MicroVAX system is the only VAX computer that uses a Q-bus as its interconnect.
- **VAX Backplane Interconnect (VAXBI)** is a versatile bus that connects peripherals to the system.
- **MASSBUS** equipment serves as the interconnect for most high-speed, high-density disk and tape drives (for example, TU78 tape drives).
- The VAX 8800 series uses an interconnect bus (NMI), not the VAXBI, to share memory between CPUs. The VAX 8300 series uses the VAXBI.

Different VAX processors support different buses. Some VAX processors support several buses, others support only one. Furthermore, different buses support different peripherals. For example, to connect a certain disk drive to your system, the system must have the bus required to connect the disk drive to it. Some special adapters exist to allow devices that require a particular bus to be connected to a different bus.

For more information on the types of devices that are used with particular buses, refer to the *VAX Systems and Options Catalog*.

1.6 Peripheral Devices

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

1.6.1 Terminals

You can use a terminal to communicate with the computer. There are two types of terminals: hard-copy and video. Both types of terminals have keyboards for entering information. Figure 1–2 shows examples of both hard-copy and video terminals.

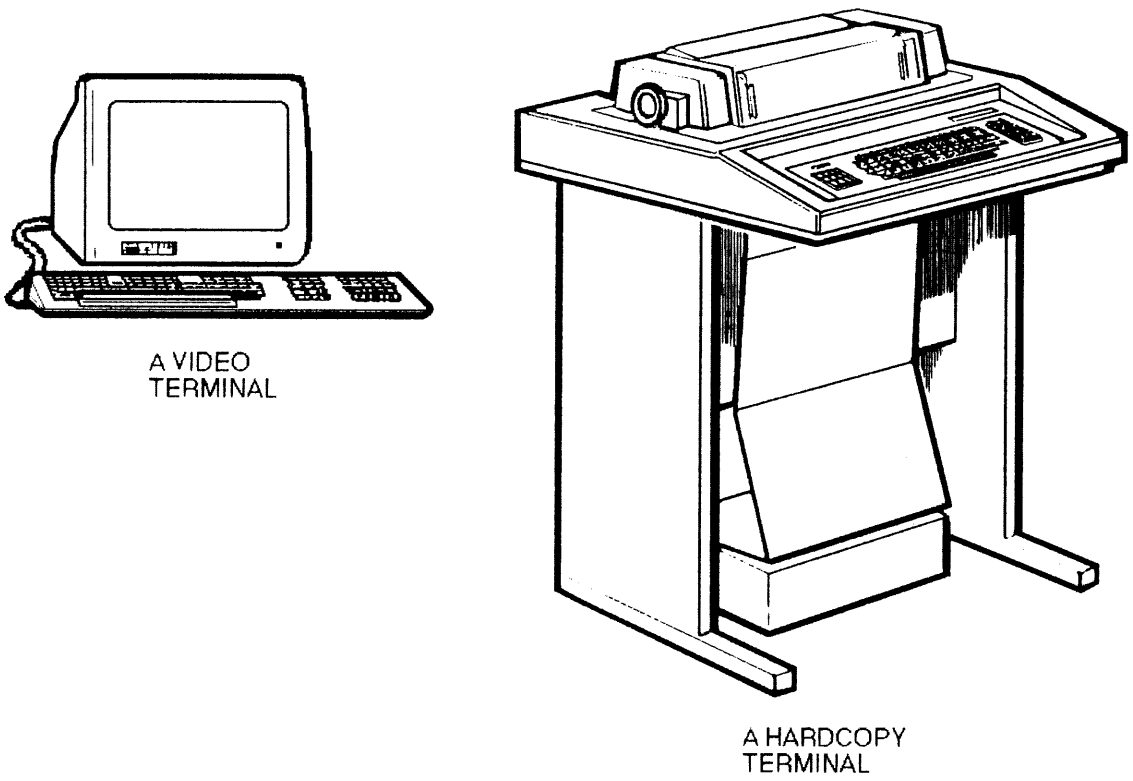


Figure 1–2 Hard-copy and Video Terminals

For information on the terminals offered by Digital, refer to the *Terminals and Printers Handbook*. For information on operating a specific terminal, refer to the *User's Guide* for the terminal.

1.6.2 Printers and Printer/Plotters

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

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

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

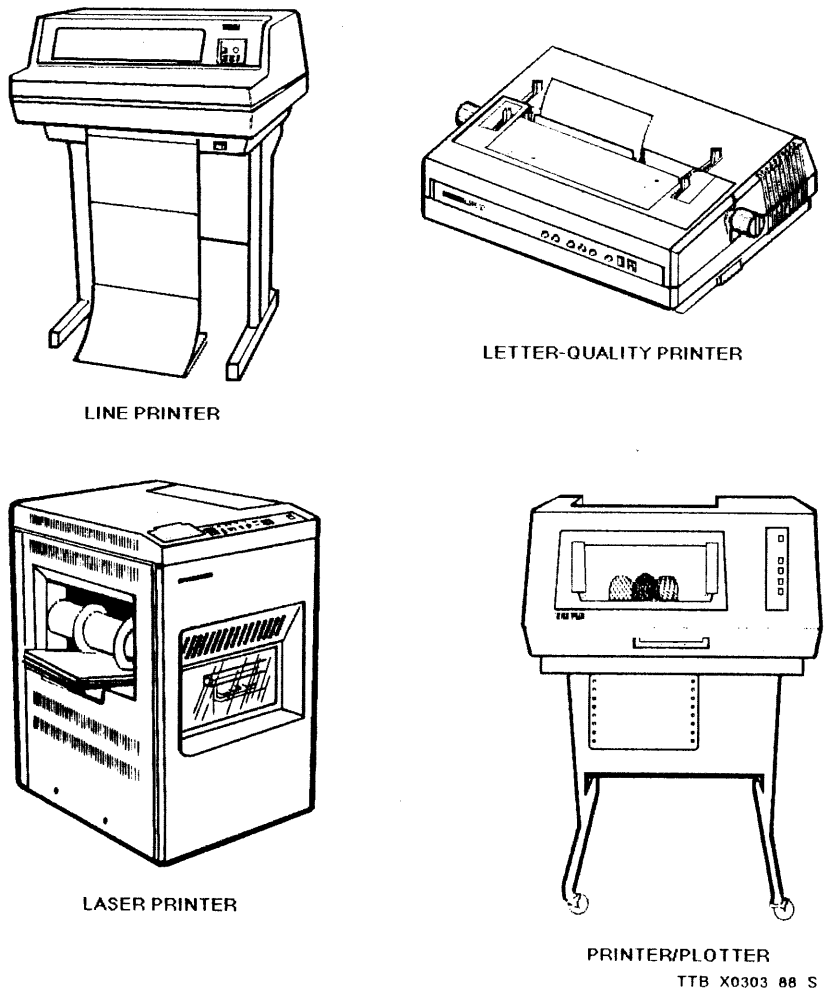


Figure 1–3 Printers and Printer/Plotter

For more information on the printers available from Digital, refer to the *Terminals and Printers Handbook*. For information on operating a particular printer, refer to the *User's Guide* for the printer.

1.6.3 Disk and Tape Drives

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

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

- Cartridges
- Disk packs
- Diskettes
- Optical disks

Each type of disk is used with a particular disk drive. For example, cartridge disks are used in RC25 disk drives, disk packs are used in RA60 drives, and diskettes are used in RX02 drives. Some disk drives have fixed disks, which means that you cannot remove the disk from the drive. The RA81 is an example of a fixed disk drive. Others have a removable disk, which means that you can remove the disk from the drive. The RA60 is an example of a removable disk drive.

A new type of disk, called an **optical disk**, is becoming more widely used on VMS systems. Optical disks use laser beams to write and read tiny spots that represent binary information. Currently, there are two primary types of optical disks:

- Read-Only (CDROM)
  - — Commonly called "compact disks"
  - — Mastered by the manufacturer, read by the user
  - — Small format (5.25 inches in diameter)
  - — Typically used to provide large amounts of text information, such as reference manuals, abstracts, etc.

- Write-Once, Read-Many (WORM)
  - — Commonly called "optical disks"
  - — Available in both small and large (12 and 14 inch) format
  - — Each spot on the disk can be written one time, then read many times
  - — Typically used for archival data storage, and large volume storage of data with slow retrieval requirements

Refer to the *User's Guide* for the appropriate disk drive for information on handling, loading, and unloading removable disk media. Figure 1–4 shows examples of various types of disks. Figure 1–5 shows some examples of disk drives.

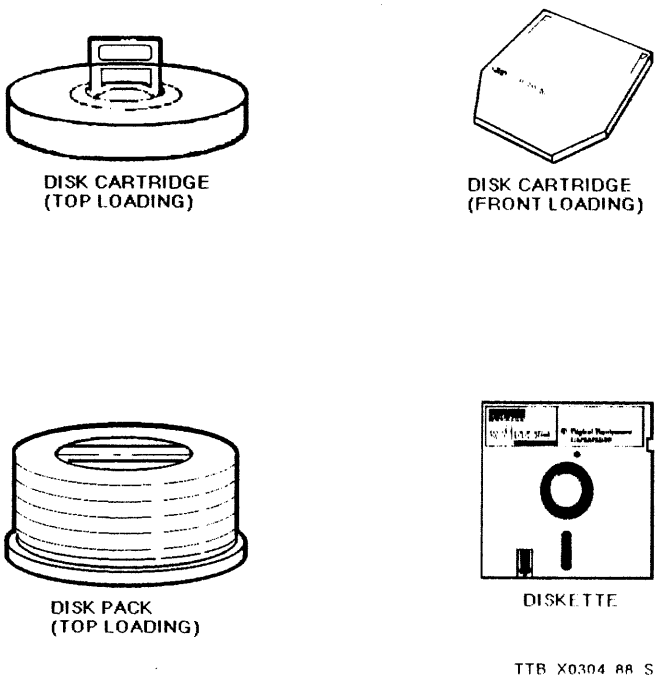


Figure 1–4 Disks

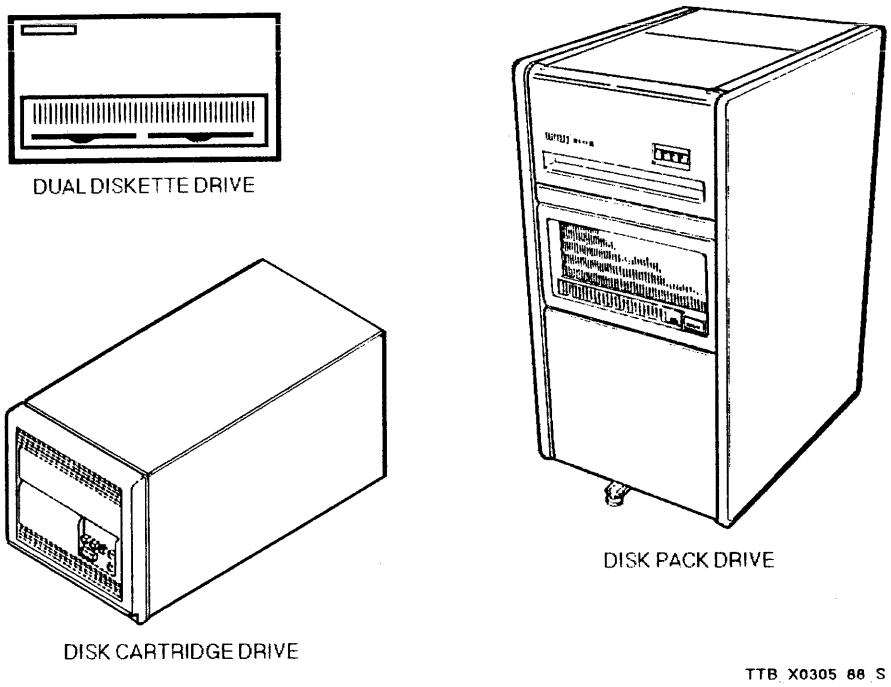


Figure 1–5 Disk Drives

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

- Reel
- Cartridge

Reel tapes come in various lengths. Longer tapes can store more data. However, it is not just the length of the tape that determines how much data it can store. Data can be recorded on tapes at different densities. The higher the density, the more data that can be stored on the tape.

The density of the data stored on the tape depends on the tape drive that is used. For example, TE16 tape drives can record data at a density of 1600 bits per inch, and TU78 drives can record data at either 1600 or 6250 bits per inch.

Tape cartridges resemble cassette tapes. The TU58 is an example of a tape cartridge. It is used as the console medium for VAX-11/750 systems. It is important to note that VMS systems organize data on TU58 cartridges similar to the way it is organized on disks. (This is not necessarily true for other tape reels and cartridges.)

For more information on operating a specific disk or tape drive, refer to the *User's Guide* for the particular drive. Figure 1–6 shows examples of tapes, and Figure 1–7 shows some representative tape drives.

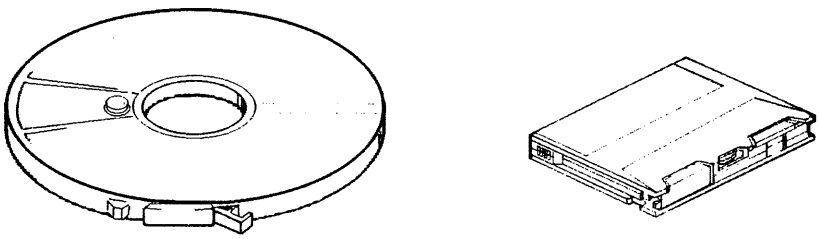


Figure 1–6 Tape Media

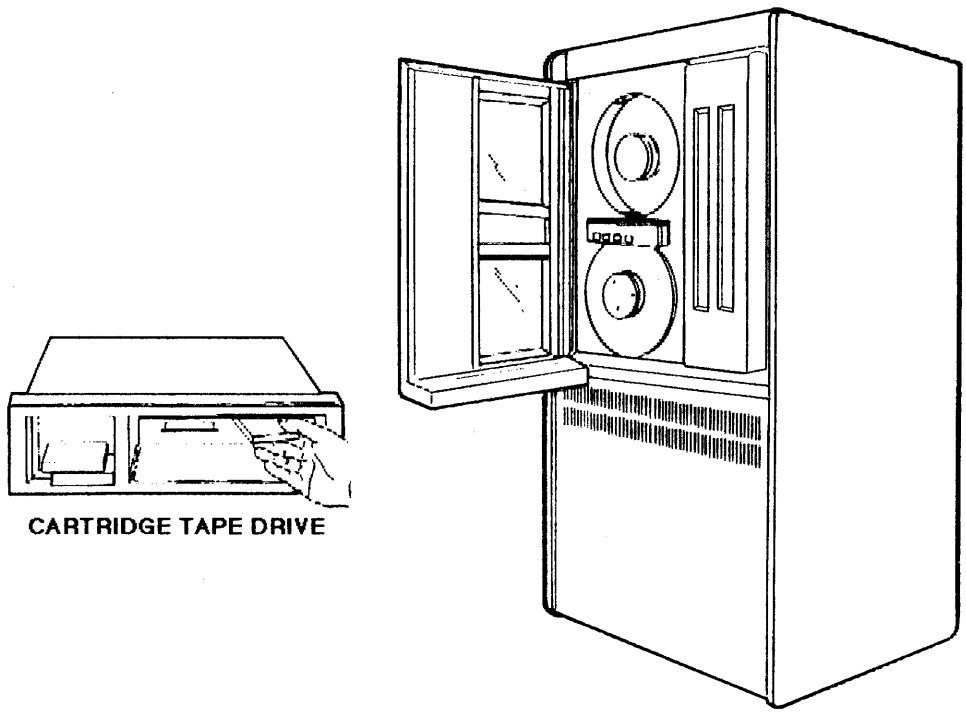


Figure 1–7 Tape Drives

1.6.4 HSC Controllers

A **hierarchical storage controller** (HSC) is an intelligent disk and tape controller. It performs all disk I/O operations, and allows data to be shared among processors in a VAXcluster system. (A VAXcluster system is described later in this module.) HSC controllers can also be used with single VAX systems. Certain disk and tape drives can be connected to the HSC, allowing the HSC, rather than the VMS system, to control and manage them.

The HSC50 and HSC70 are examples of HSC controllers. They support devices such as the RA81 disk drive, the TA78 tape drive, and the ESE20 solid state disk. For more descriptive information about HSC controllers, refer to the *VAX Systems and Options Catalog.* For information on the HSC control panel, refer to the *HSC User Guide.*

1.7 VMS Device Names

Many of your tasks will require you to specify the name of a device. All devices on VMS systems have a unique name in the format **ddcu**, where:

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 unit number of the device.

The device code specifies the device type. Table 1–1 shows some sample device codes.

Table 1–1 Device Codes

Code	Device
CS	Console storage device
DU	RA80 or RA81 disk drive
LP	Line printer
MF	TU78 tape drive

Refer to the the *VMS DCL Dictionary* (section on F$GETDVI) for a complete list of device codes.

The hardware controller number identifies the controller to which the device is connected. The hardware controller associated with the device is represented by a letter from A to Z. The controller part of the device name is assigned by the system.

The unit number is a decimal number that indicates the position of the device on the controller. You can change the unit number of a device by setting a button or switch on the device, or by installing a unit plug on the device. The unit number of the first device on a controller is usually 0. An example of a device name is MFA0, which represents the first TU78 tape drive on controller A.

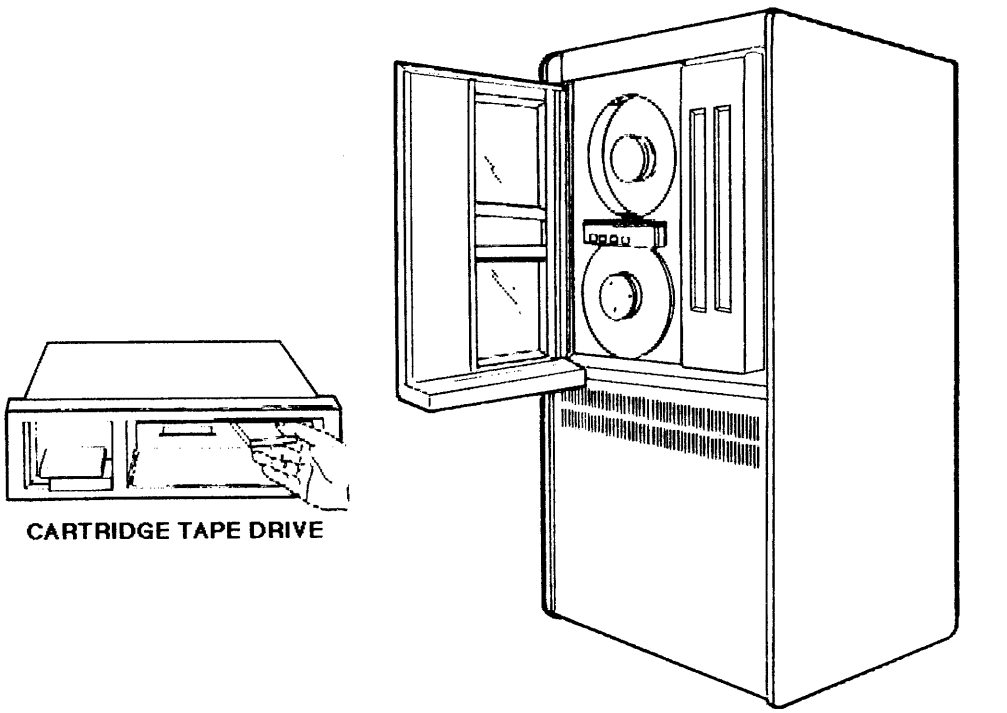


Figure 1–7 Tape Drives

1.6.4 HSC Controllers

A **hierarchical storage controller** (HSC) is an intelligent disk and tape controller. It performs all disk I/O operations, and allows data to be shared among processors in a VAXcluster system. (A VAXcluster system is described later in this module.) HSC controllers can also be used with single VAX systems. Certain disk and tape drives can be connected to the HSC, allowing the HSC, rather than the VMS system, to control and manage them.

The HSC50 and HSC70 are examples of HSC controllers. They support devices such as the RA81 disk drive, the TA78 tape drive, and the ESE20 solid state disk. For more descriptive information about HSC controllers, refer to the *VAX Systems and Options Catalog*. For information on the HSC control panel, refer to the *HSC User Guide*.

1.7 VMS Device Names

Many of your tasks will require you to specify the name of a device. All devices on VMS systems have a unique name in the format **ddcu**, where:

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 unit number of the device.

The device code specifies the device type. Table 1–1 shows some sample device codes.

Table 1–1 Device Codes

Code	Device
CS	Console storage device
DU	RA80 or RA81 disk drive
LP	Line printer
MF	TU78 tape drive

Refer to the the *VMS DCL Dictionary* (section on F$GETDVI) for a complete list of device codes.

The hardware controller number identifies the controller to which the device is connected. The hardware controller associated with the device is represented by a letter from A to Z. The controller part of the device name is assigned by the system.

The unit number is a decimal number that indicates the position of the device on the controller. You can change the unit number of a device by setting a button or switch on the device, or by installing a unit plug on the device. The unit number of the first device on a controller is usually 0. An example of a device name is MFA0, which represents the first TU78 tape drive on controller A.

1.8 Summary

- There are four main functional subsystems of VAX computers:
  - — The CPU executes instructions.
  - — The console subsystem communicates with the CPU to monitor and control the system.
  - — Main memory stores data and instructions.
  - — The I/O subsystem consists of devices, called peripherals, that provide input to and produce output from the system.
- Interconnect devices (also referred to as buses) are cables that connect the subsystems of the computer. Their most common function is to connect peripherals to the system. Common buses include:
  - — MASSBUS
  - — UNIBUS
  - — Q-bus
  - — VAXBI
- HSC controllers provide data access and disk I/O functions for a single VAX system or several VAX systems in a VAXcluster. Certain disk and tape drives can be connected to the HSC, allowing the HSC, rather than the VMS system, to control and manage them.
- There are several categories of peripherals for VAX systems. You should be able to operate the peripherals on your system, and know the type of bus used to connect it to the system. Refer to the appropriate *User's Guide* for specific information about operating each device.
- All devices on a VMS system have a device name in the format **ddcu**, where:

  dd = a two-letter device code

  c = the hardware controller (letter from A to Z)

  u = the unit number (decimal digit)

1.9 System Configurations

Given the various VAX processors, interconnects, and peripheral devices described in the first three sections of this module, you can build many different configurations. System configurations can be classified as:

- Single processors
- Multiple processor configurations

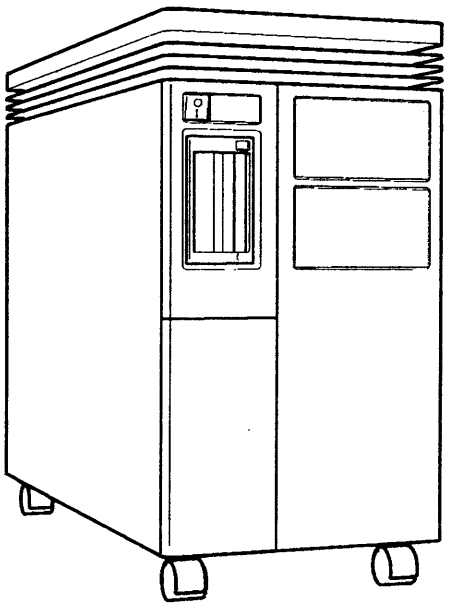
In this context, the word "system" can have different meanings. A single VAX processor, including its peripheral devices, is called a system. However, the word "system" is also used in reference to a collection of VAX processors that compose a multiple processor configuration. The following sections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

1.9.1 Single Processor Configurations

A single processor configuration is any single VAX processor and its peripheral devices. The family of VAX processors is wide ranging, and includes the following as well as other processors:

- VAX 8810
- VAX 8600
- VAX 8500
- VAX 8200
- VAX-11/780
- VAX-11/750
- MicroVAX 3000
- MicroVAX II

Note that this list reflects only some of the announced, actively marketed processors that are available at the time of this writing. For complete, up-to-date descriptions of all of the available VAX processors, refer to the current edition of the *VAX Systems and Options Catalog*. Figure 1–8 shows a MicroVAX II system, and Figure 1–9 shows a VAX 8600 system.



TTB_X0308_88

Figure 1–8 MicroVAX II System

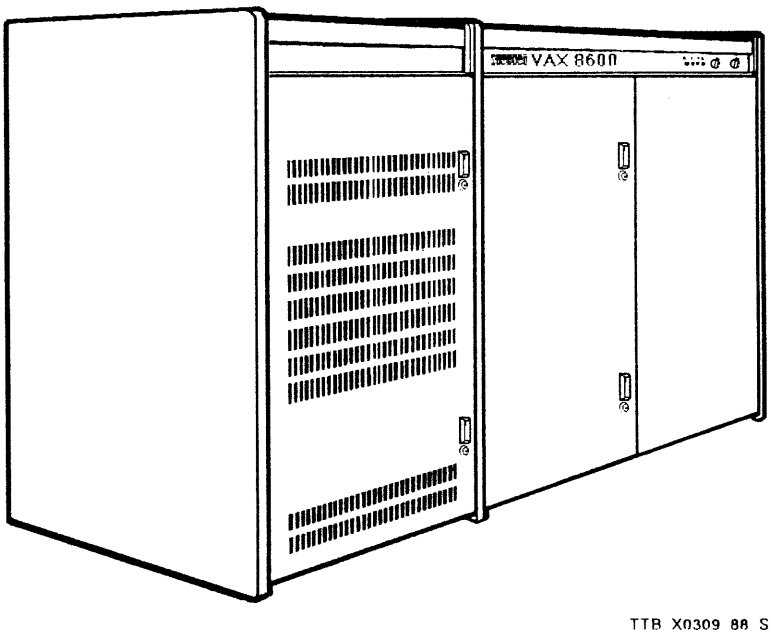


Figure 1–9 VAX 8600 System

1.9.2 Multiple Processor Configurations

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

- Tightly coupled configurations (multiprocessors)
- Loosely coupled configurations (networks)
- VAXcluster systems

In **tightly coupled** configurations, the processors share the same operating system code in the same memory; they cannot operate independently.

In **loosely coupled** configurations, each processor executes a separate copy of the operating system in its own memory; they can operate independently. This ability to operate independently is a major asset for systems requiring high availability.

VAXcluster systems are positioned midway between tightly coupled and loosely coupled configurations. Each VAX processor, or **node**, in a VAXcluster system executes its own copy of the VMS operating system. A VAXcluster is configured so that users on all nodes use a common file system and common devices. Also, a VAXcluster can continue operating without the participation of all processors. Table 1–2 summarizes the differences between tightly coupled, loosely coupled, and clustered configurations.

Table 1–2 Comparing Multiple Processor Configurations

	Tightly Coupled <————> Loosely Coupled		
System Characteristic	**Multiprocessor**	**VAXcluster**	**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 (VMS)	Separate (all VMS)	Separate (some may not be VMS operating system)
File system	Integrated	Integrated	Separate
Growth potential	Limited	Very great	Very great

These different multiple processor configurations are created by using various combinations of the VAX processors, interconnects, and peripheral devices. Multiple processor configurations are discussed in more detail in the following sections.

1.9.2.1 Tightly Coupled Configurations

A VAX 8820 series processor is an example of a tightly coupled configuration consisting of two processors that share memory by means of an NMI interconnect bus.

Tightly coupled 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 1–10 shows an example of a tightly coupled system configuration.

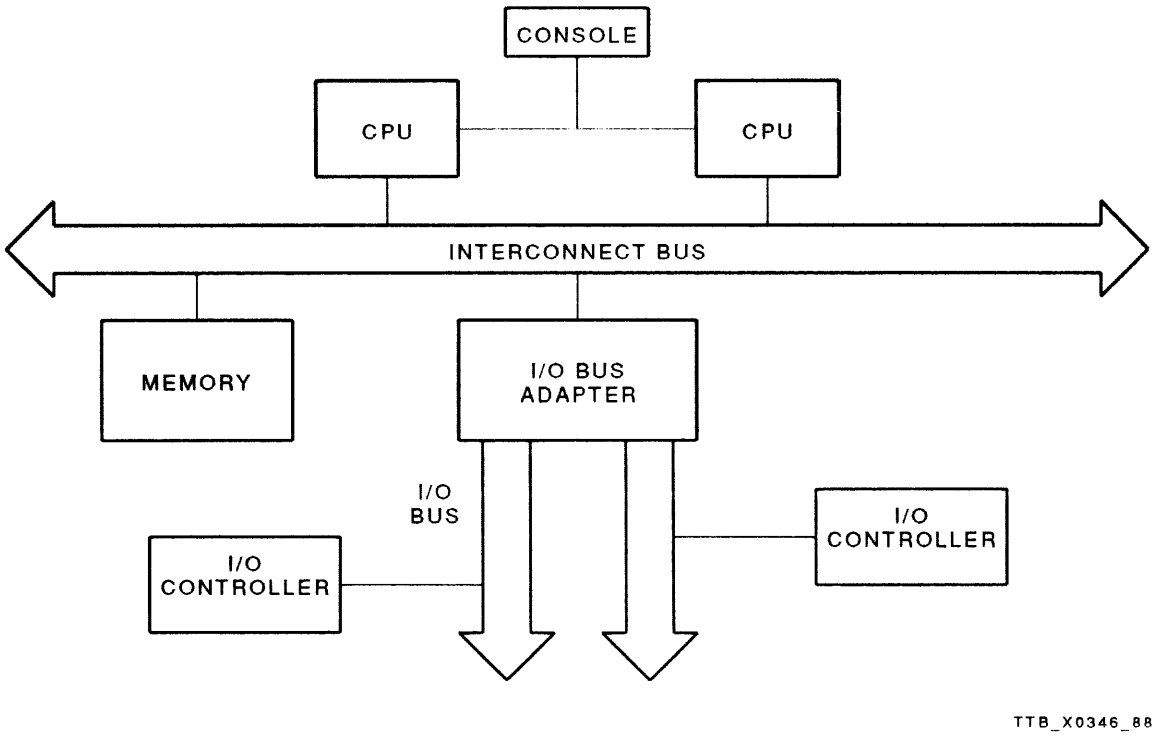


Figure 1–10 Tightly Coupled System Configuration

1.9.2.2 Loosely Coupled Configurations

A network is an example of loosely coupled systems. A network can consist of two or more communicating processors. A VMS system can be connected to other Digital systems, or to other manufacturers' systems, to establish a network. This section discusses Digital-to-Digital networks, that consist of two or more Digital systems.

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

Digital-to-Digital networks are established using:

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

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 1–11 shows an example of a DECnet network.

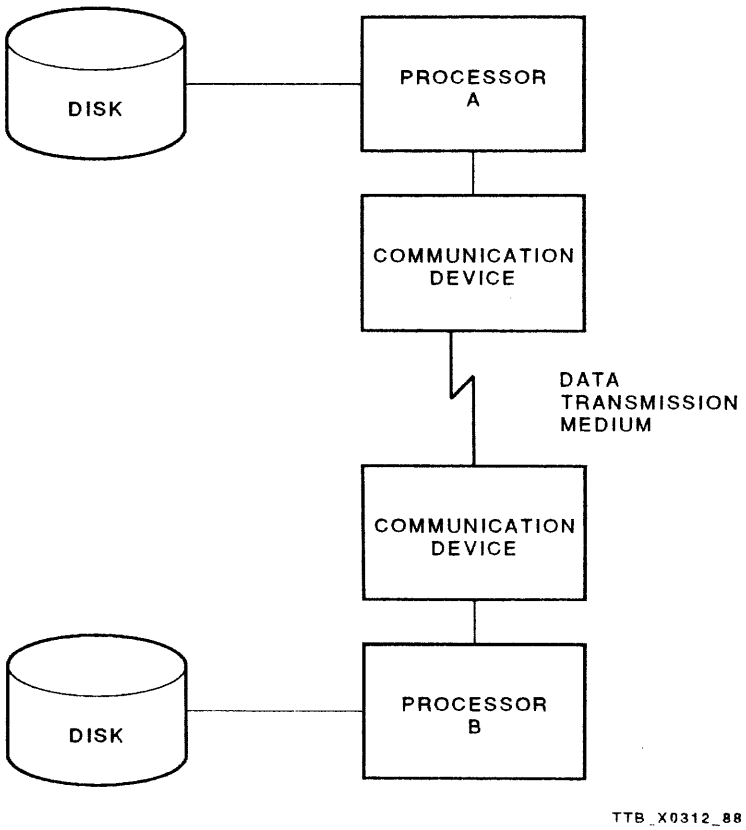


Figure 1–11 DECnet Network

Notes on Figure 1–11:

- 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. Communication devices and transmission media are discussed below.
- 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 depend upon the availability of that node. For example, if Processor A is shut down, any disk files stored on Processor A's disk become inaccessible to users logged in to Processor B.

Communication Devices

As indicated in Figure 1–11, communications hardware is needed to achieve communication between computers (or between terminals and computers). Communication between computers is usually accomplished by means of **synchronous** devices, which operate at relatively high speeds. The DEQNA and the DMR11 are examples of synchronous communication devices that are used to connect computers.

Communication between terminals and computers is accomplished by means of **asynchronous** devices, which are slower than synchronous devices. The DL11 and the DMZ32 are examples of asynchronous communication devices that are used to connect terminals to computers. Some communication devices support both synchronous and asynchronous communication; the DMF32 is an example of a synchronous/asynchronous device.

The different communication devices vary with respect to characteristics and function. For example, each communication device can be connected to a specific type of bus. For complete, up-to-date descriptions of the available communication devices, refer to the current edition of the *Networks and Communications Buyer's Guide.*

Data Transmission Media

Transmission media are required to transmit data between the communication devices attached to communicating computers. There are various types of data transmission media, including:

- Ethernet cable
- Fiber optic cable
- Telephone lines
- Satellite links

The different types of transmission media vary with respect to their transmission speeds, the distances they can span, their operating and performance characteristics, and the capabilities and services they offer.

Terminal Servers

Terminal servers connect terminals to computers in a local area network (LAN). Each terminal connected to a terminal server can access other systems connected to the same LAN. If a user's terminal is connected to a terminal server, the user can log in to other systems connected to the LAN. Therefore, terminals connected to a terminal server are more flexible than terminals connected directly to a specific processor.

Terminal servers can also be used to make printers accessible to any system on the LAN. Figure 1–12 shows the use of terminal servers in a network.

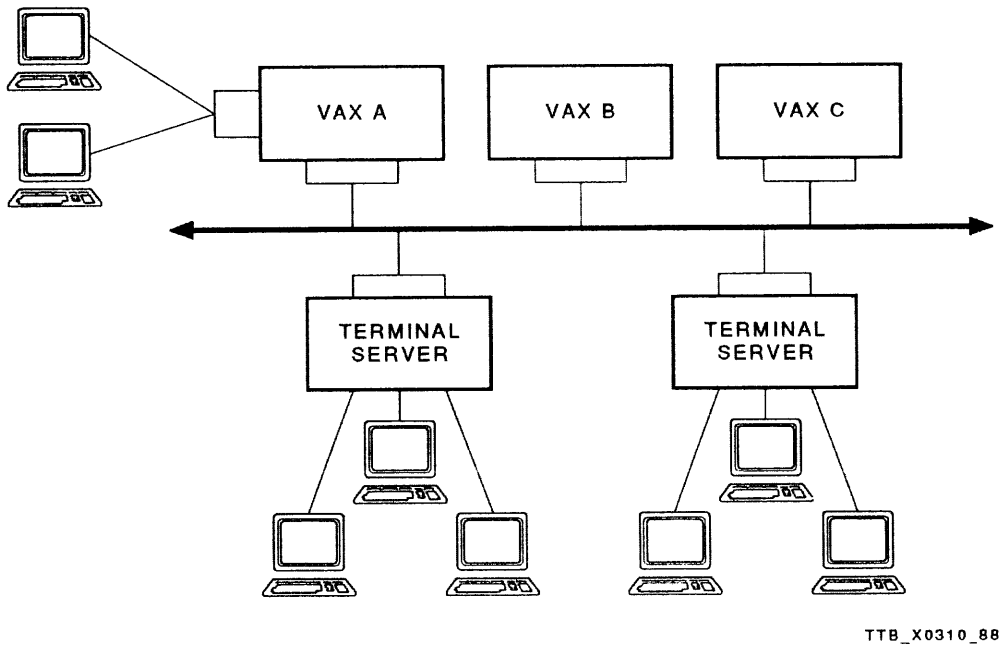


Figure 1–12 Configuration with Terminal Servers

Notes on Figure 1–12:

- The configuration includes three nodes: VAX A, VAX B, and VAX C.
- VAX A has two terminals connected directly to it. These two terminals are dependent upon VAX A; if VAX A is shut down, these two terminals become unusable.

- There are two terminal servers connected to the local area network. Any of the terminals connected to the terminal servers can be used to log in to any available node (VAX A, VAX B, or VAX C). Therefore, these terminals provide more flexibility than the terminals that are connected directly to VAX A.
- If any of the three nodes is shut down, the terminals connected to the terminal servers can still be used to log in to the available node(s).

1.9.2.3 Clustered Systems

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

However, each VMS system in a properly configured VAXcluster system boots and fails separately, as in loosely coupled configurations. Therefore, a VAXcluster system has characteristics of both loosely and tightly coupled systems.

A VAXcluster system allows the user to perform functions similar to those a network allows. The important difference between a VAXcluster system and a network is that, in addition to providing the functions of a network, a VAXcluster system provides:

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

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

As your need for computing power increases, you can increase the number of VMS systems in your VAXcluster system; as your mass storage needs increase, you can add more storage devices.

VAXcluster system hardware components include:

- VMS computer systems, composed of:
  - — VAX processors
  - — Local peripherals

- HSC storage subsystems
- Interconnection interfaces
  - — A Computer Interconnect (CI), composed of:

    A CI bus
    An SC008 Star Coupler
    An intelligent I/O port (a CI780, CI750, or CIBCI)

  - — A Local Area Network (Ethernet) connection, composed of:

    An Ethernet interface (DEUNA, DELUA, DEBNA)
    A transceiver or multiport transceiver (DELNI)
    A transceiver cable

Figure 1-13 shows a VAXcluster system utilizing a pair of HSC controllers to provide a high degree of performance and availability to the shared disks attached to the controller. Figure 1-14 shows a **local area VAXcluster** system in which a number of VAX systems are connected by way of Ethernet local area network connections.

A VAXcluster system can employ either a CI or Ethernet connection, or both. A VAXcluster system employing both CI and Ethernet connections is called a **mixed-interconnect VAXcluster** system. Figure 1-15 shows a mixed-interconnect VAXcluster system combining both a CI and Ethernet connections.

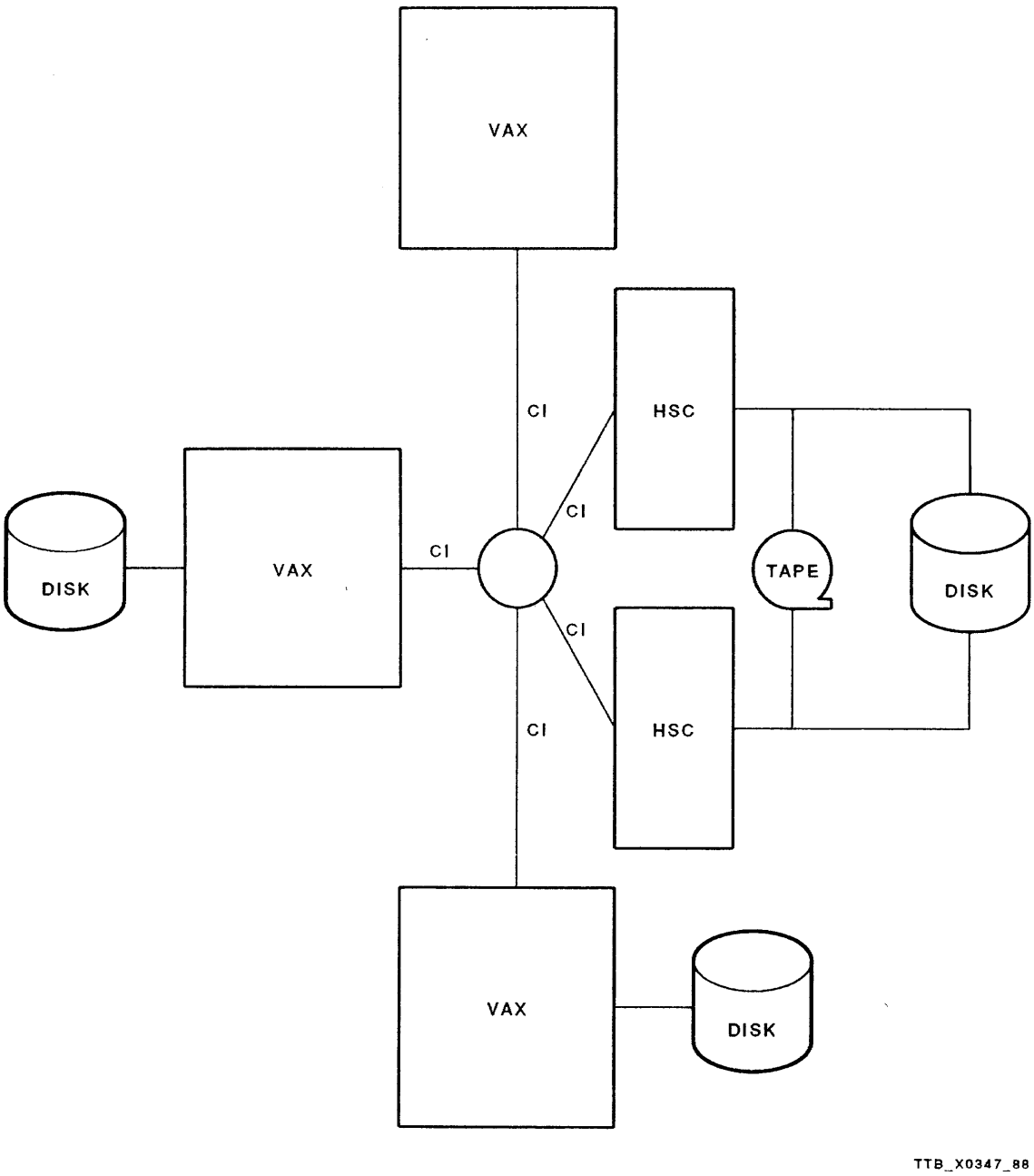


Figure 1–13 CI-Only VAXcluster System Configuration

Notes on Figure 1–13:

- The CI-only VAXcluster system configuration shows three VAX systems (two having local disks), and two HSC controllers with two dual-ported disks connected to them.
- Any of the three VAX systems in this VAXcluster system can mount the two disks that are connected to the HSC controllers. HSC disks are more available to users than local disks because HSC disks are not dependent upon the availability of any processor. The two local disks (connected directly to the two VAX systems) can be made available to any of the VAX systems, but if a VAX system with a local disk goes down, that disk is no longer available to the rest of the VAXcluster system.
- The disks are dual pathed to the HSC controllers, further increasing the disks' availability in the VAXcluster system. If one HSC fails, all traffic to the connected disks automatically switches to the second HSC.

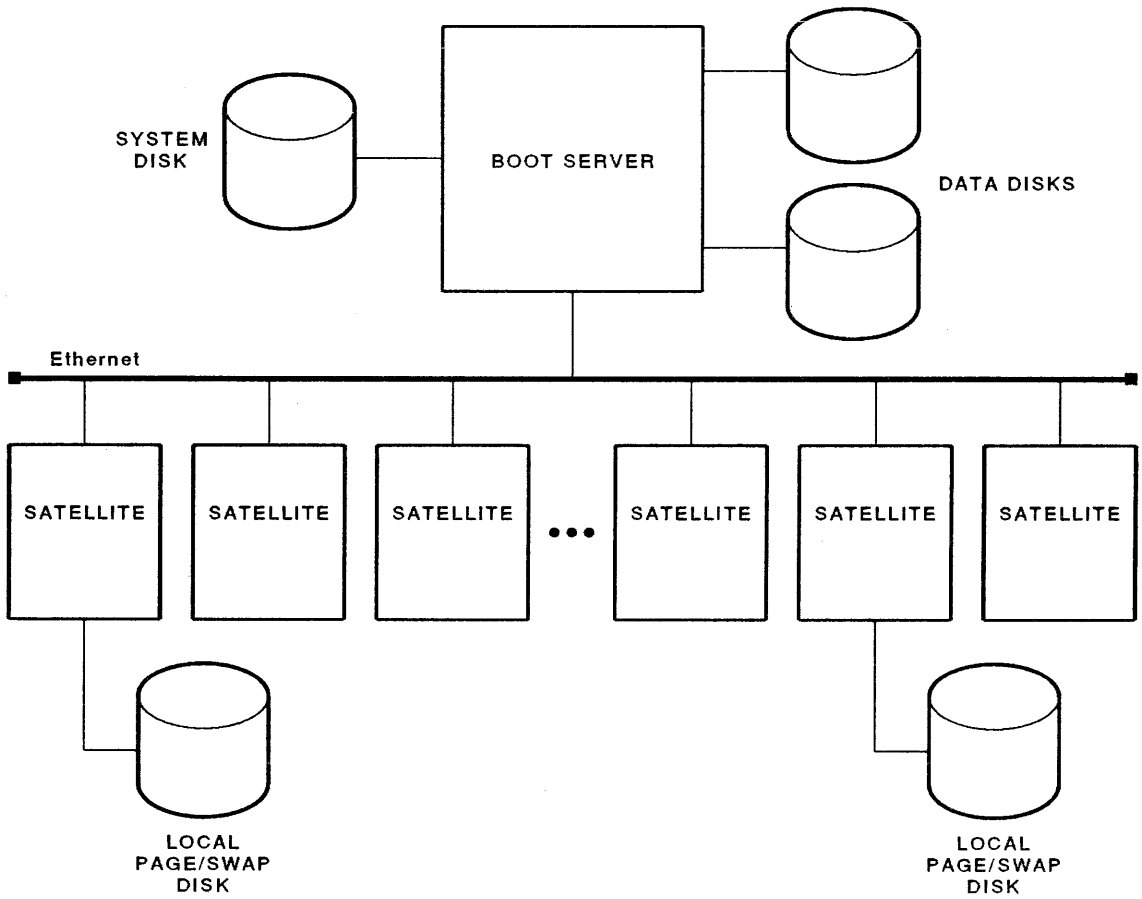


Figure 1–14 Local Area VAXcluster System Configuration

Notes on Figure 1–14:

- The local area VAXcluster system configuration shows a number of VAX systems connected by way of Ethernet local area network connections. The top VAX system functions as a disk server for the other VAX systems (called **satellites**) in the cluster.
- The top VAX system also functions as a boot server for the **diskless** satellites (those VAX systems without local disks). When a diskless satellite is booted (first started up), it requests all startup information from the boot server.

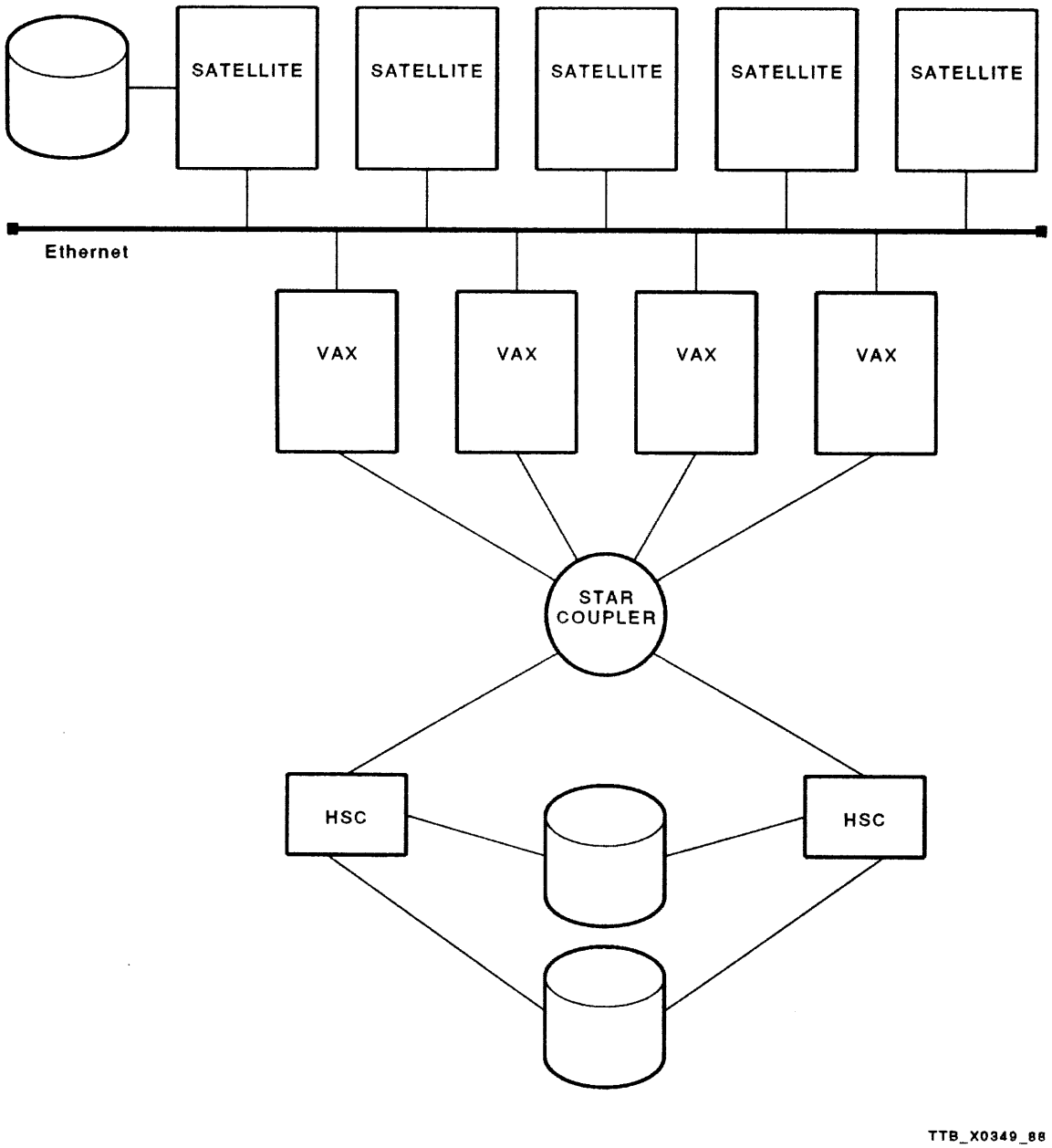


Figure 1–15 Mixed-Interconnect VAXcluster System Configuration

Notes on Figure 1–15:

- The mixed-interconnect VAXcluster system configuration shows a combination of the other VAXcluster system configurations. It is possible with this type of configuration to have all disks on each VAX system available to all other VAX systems in the VAXcluster system.
- It is important to note that in a mixed-interconnect VAXcluster system, each VAX system can have a maximum of only one CI connection and one Ethernet connection functioning as interconnection interfaces to a VAXcluster system. Additional Ethernet connections are permissible, but not as connections to the same (or a different) VAXcluster system.

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

1.10 Summary

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

1.11 Written Exercises

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

   Components of Hardware Environment:

   a. CPU

   b. Console Subsystem

   c. Main Memory

   d. I/O Subsystem

   Descriptions:

   _____ MicroVAX II processor is an example
   _____ Stores instructions and data
   _____ Used to monitor and control the system
   _____ Consists of peripherals
   _____ Executes instructions
   _____ Control panel is part of this subsystem
   _____ Used for starting up and shutting down the system

2. In the exercise below, match each device name with the appropriate description.

   Device Names:

   a. MTA0

   b. DUA2

   c. TTB4

   d. LPA0

   Descriptions:

   _____ Fifth terminal connected to controller B
   _____ First TU77 tape drive connected to controller A
   _____ Third RA81 disk connected to controller A

3. In the exercise below, match each description with the appropriate system configuration.

   System Configurations:

   a. Single processor configuration

   b. Tightly coupled system

   c. Loosely coupled system

   d. VAXcluster configuration

   Descriptions:

   _____ Nodes have separate file systems
   _____ Processors cannot operate independently
   _____ A VAX processor and its peripheral devices
   _____ An example of this type of configuration is a network
   _____ Configured midway between tightly coupled and loosely coupled systems
   _____ This type of configuration is also called a multiprocessor
   _____ All nodes reside in close proximity to each other

4. In the exercise below, match each network component with the most appropriate description.

   Network Components:

   a. Hardware communication devices

   b. Data transmission media

   c. Terminal servers

   d. DECnet software

   Descriptions:

   _____ Enables communication between networked systems
   _____ An example is Ethernet cable
   _____ Provides connections between user terminals and systems in a network
   _____ Necessary to achieve physical communication between computers
   _____ An optional component that enhances flexibility
   _____ Transmits data between communication devices

5. Indicate the letter of the best response to each of the following multiple choice questions.

_____ are used to connect the various subsystems of the computer.

a. Peripheral devices

b. Network communication devices

c. Interconnect devices

d. Storage devices

_____ have a television-like screen for displaying information.

a. Hard-copy terminals

b. Video terminals

c. Laser printers

d. Mass storage devices

A _____ is NOT a peripheral device.

a. Terminal

b. Printer

c. CPU

d. Disk drive

_____ are high-speed machines usually used for large quantities of output.

a. Hard-copy terminals

b. Disk drives

c. Laser printers

d. Line printers

_____ is NOT a type of disk.

a. Reel

b. Cartridge

c. Diskette

d. Disk pack

_____ record data on magnetic tape.

a. Disk drives

b. Tape drives

c. Terminal servers

d. VAXcluster systems

The _____ is an example of a tightly coupled system.

a. MicroVAX II

b. VAX 8820

c. VAX 8600

d. VAX-11/780

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

a. Loosely coupled system

b. Wide area network

c. Local area network

d. Multiprocessor

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

a. VAXcluster configurations

b. Wide area networks

c. Tightly coupled systems

d. Terminal servers

The major difference between a cluster and a network is _____.

a. VMS DECnet software

b. VMS cluster software

c. Communication devices

d. Processors

1.12 Solutions to Written Exercises

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

   Components of Hardware Environment:

   a. CPU

   b. Console Subsystem

   c. Main Memory

   d. I/O Subsystem

   Descriptions:

   a MicroVAX II processor is an example
   c Stores instructions and data
   b Used to monitor and control the system
   d Consists of peripherals
   a Executes instructions
   b Control panel is part of this subsystem
   b Used for starting up and shutting down the system

2. In the exercise below, match each device name with the appropriate description.

   Device Names:

   a. MTA0

   b. DUA2

   c. TTB4

   d. LPA0

   Descriptions:

   c Fifth terminal connected to controller B
   a First TU77 tape drive connected to controller A
   b Third RA81 disk connected to controller A

3. In the exercise below, match each description with the appropriate system configuration.

   System Configurations:

   a. Single processor configuration

   b. Tightly coupled system

   c. Loosely coupled system

   d. VAXcluster configuration

   Descriptions:

   ___c___ Nodes have separate file systems
   ___b___ Processors cannot operate independently
   ___a___ A VAX processor and its peripheral devices
   ___c___ An example of this type of configuration is a network
   ___d___ Configured midway between tightly coupled and loosely coupled systems
   ___b___ This type of configuration is also called a multiprocessor
   ___d___ All nodes reside in close proximity to each other

4. In the exercise below, match each network component with the most appropriate description.

   Network Components:

   a. Hardware communication devices

   b. Data transmission media

   c. Terminal servers

   d. DECnet software

   Descriptions:

   ___d___ Enables communication between networked systems
   ___b___ An example is Ethernet cable
   ___c___ Provides connections between user terminals and systems in a network
   ___a___ Necessary to achieve physical communication between computers
   ___c___ An optional component that enhances flexibility
   ___b___ Transmits data between communications devices

5. Indicate the letter of the best response to each of the following multiple choice questions.

__c__ are used to connect the various subsystems of the computer.

a. Peripheral devices

b. Network communication devices

c. Interconnect devices

d. Storage devices

__b__ have a television-like screen for displaying information.

a. Hard-copy terminals

b. Video terminals

c. Laser printers

d. Mass storage devices

A __c__ is NOT a peripheral device.

a. Terminal

b. Printer

c. CPU

d. Disk drive

__d__ are high-speed machines usually used for large quantities of output.

a. Hard-copy terminals

b. Disk drives

c. Laser printers

d. Line printers

__a__ is NOT a type of disk.

a. Reel

b. Cartridge

c. Diskette

d. Disk pack

__b__ record data on magnetic tape.

a. Disk drives

b. Tape drives

c. Terminal servers

d. VAXcluster systems

The __b__ is an example of a tightly coupled system.

a. MicroVAX II

b. VAX 8820

c. VAX 8600

d. VAX-11/780

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

a. Loosely coupled system

b. Wide area network

c. Local area network

d. Multiprocessor

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

a. VAXcluster configurations

b. Wide area networks

c. Tightly coupled systems

d. Terminal servers

The major difference between a cluster and a network is __b__.

a. VMS DECnet software

b. VMS cluster software

c. Communication devices

d. Processors

1.13 Review of VMS System Concepts

Before discussing the user software and process environments, a quick review of VAX hardware and VMS operating system concepts is presented, which provides a framework for understanding later sections. The overview includes these topics:

- DCL and the VMS Operating System
- Programs, Images, and Utilities
- Processes
- Hardware and Software Contexts
- Virtual Address Space
- Working Sets and Balance Set
- Paging
- Scheduling
- Swapping
- Processor Access Modes

Most of the material in this section is detailed in the video-based course titled VAX Concepts, or the lecture/lab course titled VMS System Architecture. If you have taken this course, or feel you have sufficient understanding of these topics, you may want to skip this section. If after reading these review topics you feel unsure of the material, you are encouraged to examine the VAX Concepts course.

1.13.1 The DIGITAL Command Language (DCL) and the VMS Operating System

The **VMS operating system** enables you to work easily and effectively. Like any operating system, it must perform three major functions:

- Provide the means for you to communicate with the devices that are part of your installation.
- Create a working environment for you, allowing you to use the resources of the system while preventing you from interfering with the activities of other users.
- Schedule the use of the CPU, physical memory, and peripheral devices to give you fair service, while it keeps each of these resources as busy as possible.

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 them. The CLI calls the VMS operating system routines that execute your request.

1.13.2 Programs, Images, and Utilities

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 linked to become a machine-executable **image**, as illustrated in Figure 1–16.

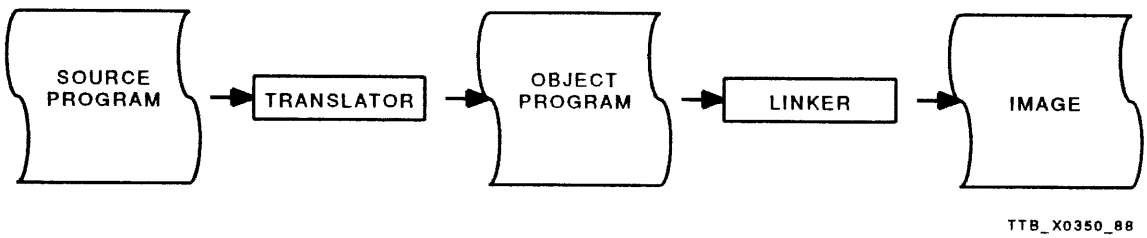


Figure 1–16 The Process of Translating a Program into an Image

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 we will henceforth use "program" in the same ambiguous manner. For system management purposes, this ambiguity does not really present a problem, since for the most part, system managers deal with executable images and not source programs.

Programs typically execute the commands you specify at your terminal. The VMS operating system includes a large number of programs that help you perform your data processing task quickly and effectively. You use many of these programs when you enter VMS 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 VMS operating system are often called **utilities**. Some utilities, such as text editors, have command languages of their own. To use these utilities, you must first use the DCL command that calls the utility, and then use the utility's own command language.

1.13.3 Processes

In the VMS operating system, images execute in the context of a **process**. A process is the total environment in which an image executes. After an image completes execution, the process continues to exist, providing an environment for the next image to execute. A process is the fundamental entity that VMS systems schedule for execution, not just the image within the process.

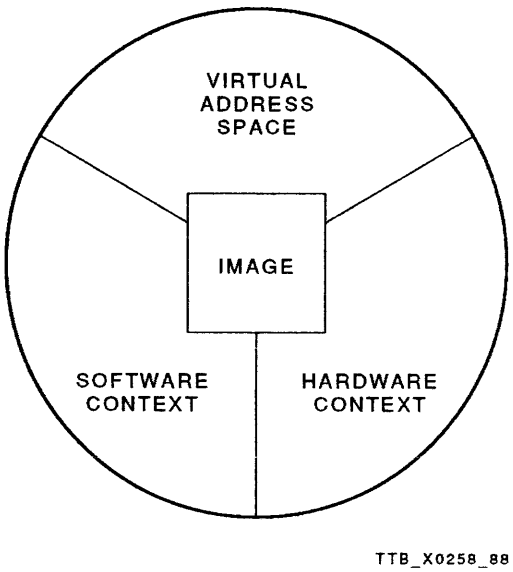


Figure 1–17 Components of a VMS Operating System Process

As shown in Figure 1–17, a process is composed of three parts: a virtual address space, a hardware context, and a software context (together, called the **context**). Whenever the VMS operating system preempts (temporarily halts) execution of a process (for example, to allow other processes in the system to execute), it saves the information needed to accurately restore the process to its previous execution state. This operation is known as **context switching**.

1.13.4 Hardware and Software Contexts

The **hardware context** refers to the state of the VAX processor at any given moment, specifically the contents of the processor registers and the processor status longword (PSL). By saving the hardware register contents (the hardware context), the system allows another process to execute, and later restore the original process' hardware context. In this way, the original process can continue execution at exactly the point at which it was interrupted.

The **software context** is analogous to the hardware context in that it describes the software status of a process rather than the hardware status. The software context includes such things as a unique process identification number (PID), accounting information, priority, privileges, and quotas. Together, these software status components define what a process can and cannot do. For example, the software context defines how much memory, how much processor time, and which devices a process is allowed to access.

An important aspect of VMS system management is the granting and management of the priorities, privileges, and quotas associated with processes. This topic is discussed in Module 2.

1.13.5 Virtual Address Space

An important aspect of the environment in which an image executes is the **virtual address space** that contains the range of virtual addresses referenced by the image. Recall that a **virtual address** is symbolic or logical representation of a physical memory location in the hardware.

The acronym "VMS" stands for "**V**irtual **M**emory **S**ystem." Put simply, a virtual memory system provides a way for a program to be written as if it had a very large number of contiguous memory locations available for its use. The VMS operating system, together with the VAX system hardware, manage the translation of **virtual** memory addresses to **physical** memory locations in a manner transparent to the image executing within the process. During the life of a process, the VMS operating system manages the virtual address space requirements of the process as various images execute within it.

The part of the VMS operating system that performs tasks relating to managing memory requirements is called the **memory management subsystem**. The memory management subsystem consists of three components, each of which will be discussed in detail.

- Page fault handler — manages memory on a per-process basis
- Modified page writer — assists in maintaining system page lists
- Swapper — manages memory on a system-wide basis

1.13.6 Working Sets and Balance Set

Managing virtual memory involves moving pieces (called **pages**) of the process' image to and from auxiliary storage (typically disk storage) as the image references its virtual address space. Since there is always a limited amount of physical memory (and many processes competing for it), the VMS system allows only a certain amount of the image's virtual memory to reside in physical memory at any given time.

1.13.6.1 Working Set

The amount of resident (physical) memory used by a process is called the process' **working set**. The maximum amount of physical memory a process can use is defined by the **working set quota**, an important quota managed by the VMS system manager. Different processes are given different working set quotas depending on their individual requirements and available system resources.

1.13.6.2 Balance Set

The set of processes whose working sets are resident in physical memory at any given time is called the **balance set**. The term originates with the operating system's need to balance the limited store of physical memory against the demands of many user processes (a demand usually greater than the amount of physical memory). A process whose working set is in memory is in the balance set. Conversely, a process swapped out (described shortly) is not in the balance set. The balance set is defined system-wide, since memory is a system-wide resource and the operating system defines the allocation of physical memory. The **balance set** of the **system** is analogous to the **working set** of a process.

1.13.7 Paging

Whenever an image references a page of its virtual memory that is not currently resident in physical memory, a **page fault** occurs. When this happens, the VMS operating system temporarily suspends the process while it fetches the referenced page. This operation is called **paging**, and the operating system routine that handles paging is called the **page fault handler**.

If a page has never been referenced, it is still located in the **image file** in auxiliary storage. Remember that an image file is created by the linker from **object files**, and that object files are created by a compiler (translator) from source programs. Every page of executable machine-language code is in the image file before it enters the process working set. A page of code (or data) enters the working set only when actually referenced. This is called **paging on demand** (or **demand paging**), since paging is invoked only when a page is demanded by an image actually referencing it. In time, pages are faulted into the working set as the flow of control moves through the image.

1.13.7.1 Working Set Extent

Often, as an image executes within a process, it faults in so many pages that it reaches the limit of its working set quota. When this happens, the VMS operating system checks yet another process memory quota called the **working set extent**. The working set extent provides the process with a sort of "overdraft," allowing an image to fault in a limited number of additional pages. The working set extent can have different values for each user; it is an important quota controlled by the VMS system manager.

When an image faults so many pages that even the process working set extent is exhausted, the VMS operating system forces the process to surrender enough pages to make room for the new pages. Which pages are surrendered is determined by the **working set list**, the ordered list of pages in the process' working set. One can view the working set list as a queue structure in which a new page is added at one end and the oldest page is taken off at the other end. The VMS page fault handler essentially uses the working set list to identify a page in the working set that has not been recently referenced. Such pages are surrendered when the process needs additional physical memory space for new pages.

1.13.7.2 Free Page List

When a process needs additional physical pages, the system fetches them from the **free page list**. The free page list is a shared, system-wide pool of physical memory pages available to any process that needs another page of storage. Similarly, when a process must surrender a page (to fault in a newly referenced page), the old page is placed at the bottom of the free page list. The page gradually moves to the top of the free page list as processes request additional pages or surrender pages to the bottom of the list. If the page reaches the top, that page is the next one to be given away to the next process requesting a new page.

Should a process surrender a page, then reference that same page a short time later, the page fault handler retrieves the referenced page from the free page list page. If it finds it, the page is taken off (**dequeued** from) the free page list and placed back into the process' working set list. If the referenced page is not found on the free page list (due to another process acquiring the page), the page fault handler must go out to the image file, retrieve a copy of the page, and place it into the process' working set list.

1.13.7.3 Modified Page List

The procedure just described works for pages that have not been modified (written into) by the process. If a surrendered page has been modified, the page fault handler places the modified page on the **modified page list**, a structure similar in design and operation as the free page list.

To optimize memory management tasks, the memory management subsystem uses the free page list to quickly provide pages to processes when they need it. To maintain this optimization, the memory management subsystem tries to keep the free page list at a certain minimum size. If the free page list falls below a certain minimum limit (a parameter controllable by the VMS system manager), the memory management subsystem uses a special procedure for acquiring additional pages.

When the free page list falls below the minimum limit, a memory management subsystem component called the **modified page writer** comes into play. The modified page writer copies some of the pages in the modified page list to a **paging file** in auxiliary storage, then places those pages at the bottom of the free page list. The pages transferred from the modified page list to the free page list still contain information useful to one or more processes. Therefore, if a process references one of those pages, the page fault handler will locate the modified page in the free page list and place it back into the process' working set.

If the page has already been given to another process, the page fault handler goes out to the paging file and retrieves a copy of the referenced modified page. The relationship between the image file, free page list, modified page list, and paging file is illustrated in Figure 1–18.

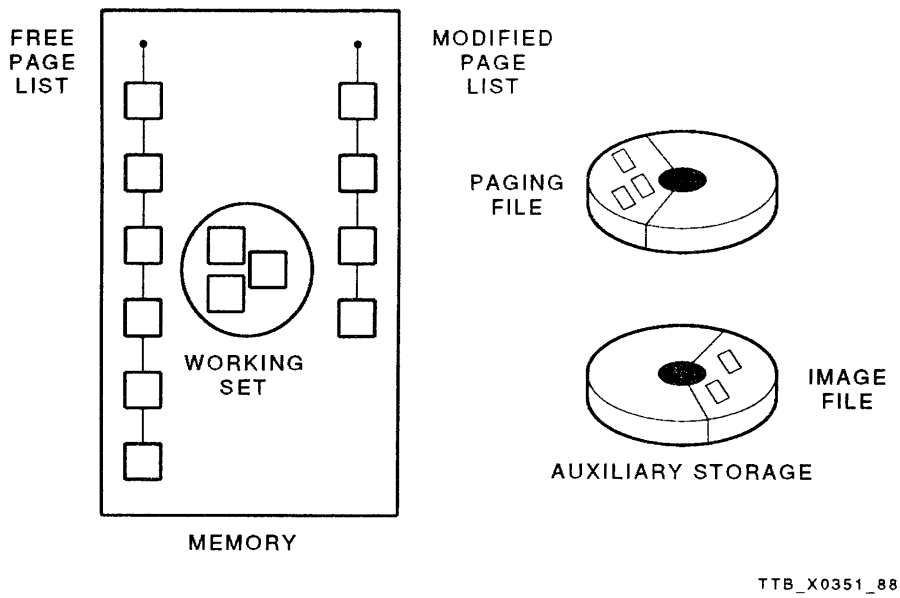


Figure 1–18 Pages in Memory and in Auxiliary Storage

The modified page list has the same "cache" effect as the free page list. A page that has been written into once is likely to be written into again. If that happens while the page is still on the modified page list, the page is faulted back into the working set without a read from auxiliary storage. The paging file provides a backup copy in auxiliary storage, as the image file does in relation to the free page list.

1.13.8 Scheduling

VMS is a **multiprogramming** operating system. (Another name for this type of operating system is called **timesharing**.) This means the operating system is designed to allow each process to run as if it had the entire system to itself. Since the processor can only execute instructions for one process at any given moment, the operating system **schedules** processes for execution by the processor. The VMS system routine that handles the assignment of priorities to processes, and determines which process should execute next, is called the **scheduler**.

The VMS system schedules processes for execution by using a combination of two scheduling techniques:

- **Priority scheduling** — a process executes until it is preempted by a higher priority process.
- **Time-sliced** — a process executes until its **time slice** (allocated processor time) expires, at which time another process of the same priority is scheduled for execution.

1.13.8.1 Priority Levels

The VMS operating system is a priority-driven operating system. There are 32 software priority levels, divided into two groups of 16 levels each (see Figure 1–19).

- **Real-time priority levels** — The upper 16 levels (16-31) are strictly priority scheduled, with a fixed priority for each process. Time-critical priorities are used for real-time applications that require immediate response to events. Real-time processes do not have time slices assigned to them; instead, they execute until they relinquish the processor, or are preempted by a higher priority process. Real-time processes of the same priority participate in round-robin scheduling (within their priority) if they are simultaneously computable. These processes are designed to use as little processor time as necessary to complete their tasks.
- **Normal priority levels** — The lower 16 levels (0-15) are priority scheduled, but use variable, not fixed priorities. Normal priority levels are used for interactive and batch applications. Normal processes are given time slices.

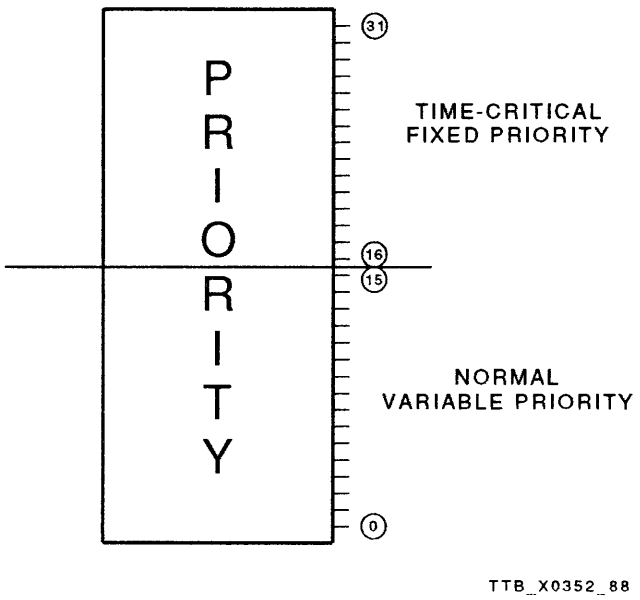


Figure 1–19 Priority Levels in the VMS Operating System

At each priority level, there is a queue of processes that wait their turn to execute. Any process in one of these priority queues must be both resident in memory (**swapped in**) as well as **computable** (executable). It is possible that some of these priority queues have no processes in them.

When a process is selected to run, because it is at the top (head) end of the highest-priority queue, it is removed from the queue. If during execution the process is preempted, it is still computable and returns to the bottom of the queue from which it came. If, however, the process has entered a **wait state** (for example, requests a disk operation), it goes to wait for the event in another queue elsewhere in the system. When the event occurs, the process returns to the bottom of the priority queue from which it originally came. By returning a process to the bottom of the priority queue, round-robin scheduling is achieved. In this way, processes at the same priority level are guaranteed a rough equality of service.

1.13.9 Swapping

So far, we have mentioned that the VMS system is a multiprogrammed (time-sharing) operating system that handles execution for multiple processes resident in the system at the same time. Furthermore, these processes require virtual memory to execute images, and the VMS operating system manages the translation of virtual memory to physical memory through the page fault handler routine.

As processes execute, they often require additional physical memory pages for their working sets. These additional pages are taken from the free page list, but at some point the size of the free page list becomes too small. For a new process to be created, or an existing one to expand, the VMS operating system needs a way to reclaim pages from existing process working sets to add to the free page list. The technique used to provide these additional free pages is called **swapping**. Swapping is the transfer of a process' working set between main (physical) memory and auxiliary storage (disk). The VMS operating system routine that handles the swapping of processes is called the **swapper**.

The VMS operating system maintains a special area in auxiliary storage called the **swapping file**. This file's sole purpose is to store **swapped out** processes. Monitoring the calculated size of the swapping file is one of the responsibilities of the VMS operating system manager.

Process state is a prime determinant of which processes get swapped in or out. If there is a high-priority process in auxiliary storage that has just become executable, the system will make room for it in memory. If necessary, lower-priority processes are swapped out to make room for the higher-priority process. This is consistent with the priority scheduling employed by the VMS operating system.

The strategy of the VMS operating system is to give high-priority processes the resources they need as soon as possible. If this requires preempting a low-priority process' execution, then so be it. The same is true if low-priority processes have to be swapped out to auxiliary storage to make room for high-priority processes.

Swapping and scheduling are distinct but closely related functions in a multiprogramming system. The strategies behind each must be compatible. In the VMS operating system, the swapper provides the scheduler with the highest-priority executable process that happens to be in auxiliary storage. The scheduler then dispatches the **highest-priority resident executable** process to execute next. If that process has a higher priority than the currently executing process, preemption occurs.

1.13.10 Processor Access Modes

Processor access modes provide a mechanism that safeguards memory from accidental or malicious actions. In addition, access modes allow the operating system to handle devices shared among a number of users, and to gain control of the system in order to schedule processes for execution.

There are four different processor access modes, shown below in decreasing order of privilege:

1. Kernel
2. Executive
3. Supervisor
4. User

Kernel mode is considered the most privileged mode, and user mode the least privileged. These access modes operate in a hierarchical fashion; that is, any capability allowed in user mode is available in more privileged modes, as shown in Figure 1–20.

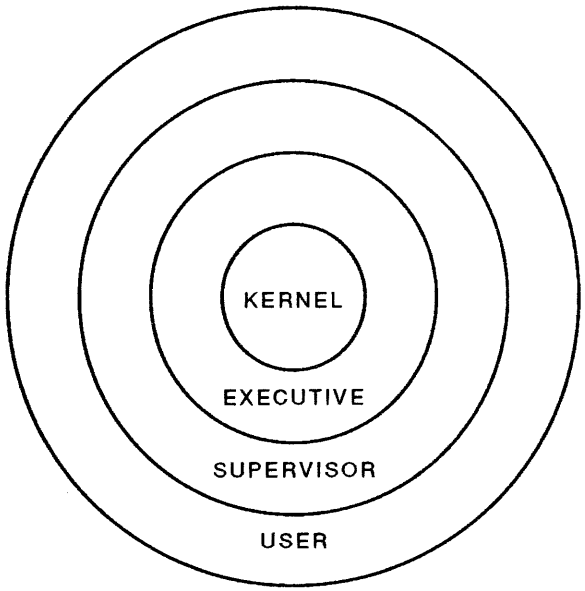


Figure 1–20 Access Mode Hierarchy

The significance of processor access modes comes into play when system-wide control operations must be performed, or when an image executing in user mode requests a privileged function. Examples of these situations include references into certain memory used by the operating system as required, when context switching must be done, or when privileged machine instructions must be executed (such as halting the processor, or responding to interrupts).

1.13.10.1 Modes Used Within the VMS Operating System

The VMS operating system does not all run in the same mode but offers a layered accessibility as shown in Figure 1–21. We can divide its components into four groups, one for each access mode:

- Components that run in kernel mode include the swapper, which swaps working sets in and out of physical memory; the pager, which responds to page faults by moving pages into the process' working set; and the routines for processing exceptions and interrupts.
- A major component that runs in executive mode is VAX Record Management Services (RMS), a package of file input/output (I/O) routines.
- Supervisor mode is used for the DCL CLI, which directly interprets the user's commands to the system.

- User mode can be used by system components that do not need any privileged access (for example, language translators, many utilities, and most user-written programs).

A process changes into a more privileged mode by executing certain system services, or by executing one of the VAX **change mode** instructions. The most commonly encountered mode change is the **change mode to kernel** situation. A process executing in user mode does not need special privileges to change to a more privileged mode (such as kernel mode) within a system service routine. Once the system service routine is entered, the routine changes to the higher mode, then resets the process back to user mode just before it returns control to the image. (The system service routine must be a privileged, installed image.)

Special privileges are needed, however, for a process to arbitrarily change to a more privileged mode when the process is in user mode. This situation requires the process to have the appropriate privileges to change into the more privileged mode, which is not a normal situation for most user processes. Again, the most common situation is when a special user process requires kernel mode execution, which is possible only if the process has the privilege called CMKRNL (change mode to kernel). Granting the CMKRNL privilege (and other privileges) represents an important system management responsibility, one that is detailed in Module 2.

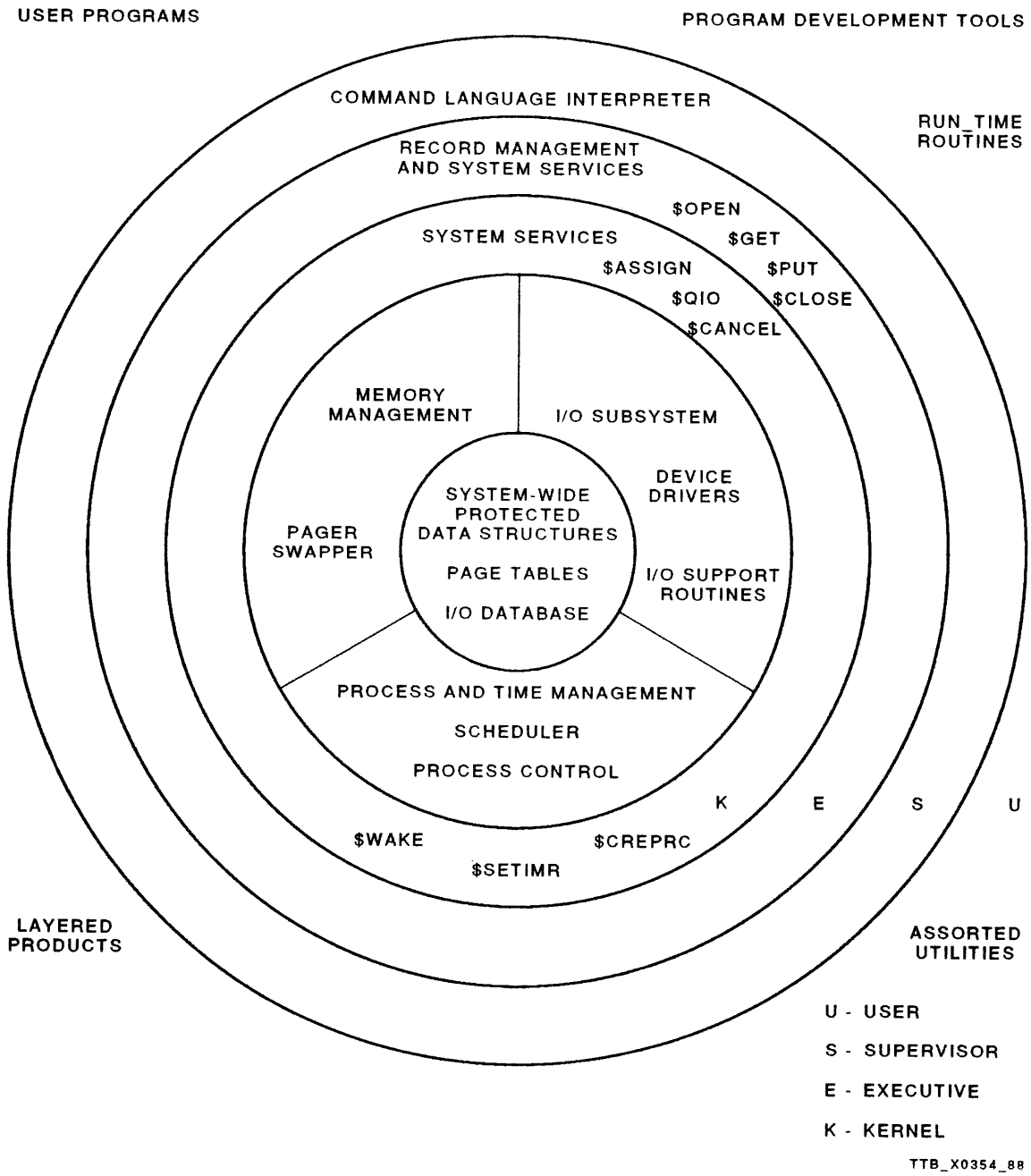


Figure 1–21 Layered Design of the VMS Operating System

1.14 System Files, Directories, and Logical Names

1.14.1 System Files

The VMS operating system stores the image files, data files, and other files it requires for its own operation on a disk volume known as the **system volume**. The disk drive on which that volume is mounted is known as the **system device**. The directories on the system volume that store the VMS operating system files are called **system directories**. Table 1–3 describes these directories and their contents. For a list of all files distributed in the system directories, see the *VMS Installation and Operation Guide* for your particular VAX processor.

Table 1–3 System Directories

Directory Name Used in Combination with SYS$SYSROOT:	Logical Name	Principal Contents
[SYSEXE]	SYS$SYSTEM	Executable images, user authorization file, site-independent startup and shutdown command procedures, system parameter files, paging file, swapping file, crash dump file
[SYSLIB]	SYS$LIBRARY SYS$SHARE	Macro libraries, object libraries, shareable image libraries, shareable images
[SYSHLP]	SYS$HELP	Help libraries
[SYSMSG]	SYS$MESSAGE	Error message files
[SYSMGR]	SYS$MANAGER	Site-specific startup and shutdown command procedures, print forms definitions, operator's log, accounting file
[SYSMAINT]	SYS$MAINTENANCE	Hardware diagnostics (not distributed with the VMS operating system)
[SYSTEST]	SYS$TEST	UETP command procedures, images, data
[SYSERR]	SYS$ERRORLOG	Error log file

Table 1–3 (Cont.) System Directories

Directory Name Used in Combination with SYS$SYSROOT:	Logical Name	Principal Contents
[SYSUPD]	SYS$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

1.14.2 System Directories

Each system directory has its own logical name, as shown in Table 1-3. Each logical name specifies the system device and a specific system directory on it. 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
```

If you want to use the directory name to refer to a file in a system directory, you must 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
```

Figure 1–22 illustrates the physical structure of the system directory tree on a VMS Version 5 system.

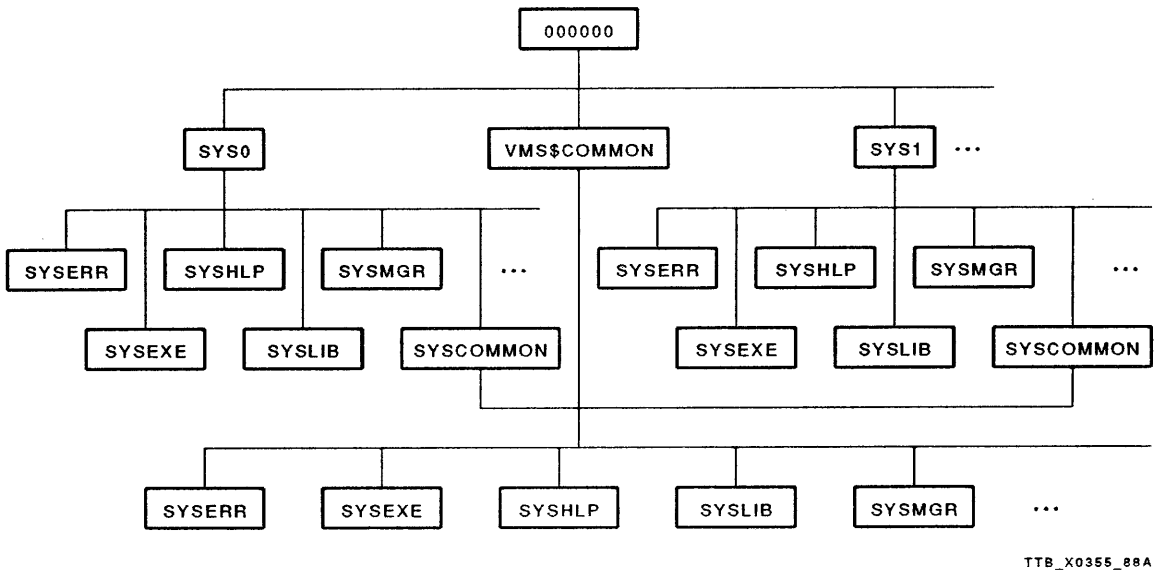


Figure 1–22 System Directory Tree

1.14.3 Concealed Root Directories

The logical name syntax for the system device and system directories implies that the logical name SYS$SYSROOT is treated like a physical device, for example DUA0. Actually, it is not that simple. If you use the command SHOW LOGICAL to translate the logical name SYS$SYSROOT, you will see that 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. In the above example, you would not be expected to include the following fully translated file specification in a command:

```
DUA0:[SYS0.SYSEXE]LINK.EXE
```

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

```
SYS$SYSROOT:[SYS0.SYSEXE]LINK.EXE
```

This example 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 in the pseudo-device DUA0:[SYS0.], i.e., SYS$SYSROOT.

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

The reason system directories are subdirectories of a first-level root directory is to allow up to 16 different versions of the operating system to coexist on the same volume. Each system version has its own root directory of the form SYS$SYSDEVICE:[SYSn].

The purpose of the logical name SYS$SYSROOT and the logical names in Table 1-3 is to conceal the existence of the first-level **root directory** [SYS0], and the fact that [SYSEXE] is actually a subdirectory of [SYS0].

Under usual operating conditions, 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 a VMS system. Because command procedures carry out an upgrade, it is unnecessary to understand concealed root directories to upgrade the system. Module 7 discusses the subject of upgrades further.

1.14.4 System File and Directory Protection

The protection of system directories and files is important for system security. System directories and files are distributed with sufficient protection to keep them safe from all users without elevated privileges. To maintain system security, you should not reduce this protection. Remember the following facts about protection:

1. 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.
2. 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. The Backup utility requires R access to directories, even when exact file names and file types are supplied.

3. You should deny W access to all sensitive directory files. If a user has W access to a directory, he or she can create a file in the directory with the same name as a file already there, and with a higher version number. You must prevent this kind of tampering.

4. When a text editor creates a new version of a file, it assigns it the protection code of the previous version. Use the SET PROTECTION command to change the protection code on the new version as needed.

   When you create a new file, the system assigns it your default protection code unless you specify a different code explicitly or with an access control list (ACL) on the directory. Use the SHOW PROTECTION command to determine the protection of a file. Module 8 discusses ACLs in more detail.

   Note that the protection of an edited file can become too strong as well as too weak. For example, system default protection denies all file access to the world category of users. However, the system login file, SYS$MANAGER:SYLOGIN.COM, must permit execute (E) access to World. If a user's process cannot read this file at login, it cannot execute the DCL commands in it. For more information about how to protect files and directories, see the command description for SET PROTECTION in the *VMS DCL Dictionary*.

1.15 Software Available with VMS Systems

By using the various development tools on your system, you can develop and run your own programs. This topic is discussed in later modules. Some of the many programs and utilities available from Digital for the VMS operating system are listed in Tables 1–4 and 1–5. The list of VMS operating system software available from Digital is continually growing, so you should always contact your Digital sales representative for a complete list of current offerings.

Table 1–4 Programs and Utilities Distributed with the VMS Operating System

Editors and Word Processing Tools	
EDT	SOS Interactive Text Editor
TPU	RUNOFF Text Formatter
EVE	SLP Batch Text Editor
Program Development Tools	
VAX MACRO Assembler	VAX Librarian
VAX Linker	VAX Common Run-Time Library
VAX Debugger	
File Manipulation Tools	
VMS Copy utility	
VMS Directory utility	
VMS Backup utility	
Device and Volume Manipulation Tools	
VMS Allocate utility	
VMS Initialize utility	
VMS Mount utility	

Table 1–4 (Cont.) Programs and Utilities Distributed with the VMS Operating System

Assorted Utilities	
File Sorting and Merging utility (SORT/MERGE)	Personal Mail utility (MAIL)
Information utilities (SHOW)	Telephone utility (PHONE)

Table 1–5 Optional Programs and Utilities Available for the VMS Operating System

Languages	
VAX APL	VAX DSM
VAX BASIC	VAX FORTRAN
VAX BLISS-32	VAX LISP
VAX C	VAX PASCAL
VAX COBOL	VAX PL/I
VAX CORAL 66	VAX RPG II
VAX DIBOL	Professional Host Tool Kit Languages
Development Productivity Tools	
VAX Code Management System (CMS)	VAX Module Management System (MMS)
VAX Language Sensitive Editor (LSE)	VAX Source Code Analyzer (SCA)
DEC Test Manager (DTM)	
Graphics Tools	
VAX ReGIS Graphics Library (RGL)	VAX DECslide Graphic Presentation utility
VAX DECgraph Plotting Package	VAX Graphics Kernel System (GKS)

Table 1–5 (Cont.) Optional Programs and Utilities Available for the VMS Operating System

Information and Data Management Products	
VAX Common Data Dictionary (CDD)	VAX Forms Management System (FMS)
VAX DATATRIEVE	VAX Terminal Data Management System (TDMS)
VAX Database Management System (DBMS)	Rdb
Office Products	
ALL-IN-1 Office Menu	
VAX DECalc Spreadsheet Calculator	
VAX DECspell	
Networks and Communications Products	
DECnet-VAX	
DECnet/SNA Gateway Products	

1.15.0.1 Additional References

For more information on the VMS operating system, see the *VMS System Software Handbook*. This publication also contains useful information about optional software.

For more information about the optional software available for your VMS system, read the following sections of the most recent *VMS Software Product Description* (SPD):

- **Description** (especially the section on **Installation**)
- **Optional Software**

For additional information concerning optional software available through Digital for VMS systems, refer to the most recent versions of the following publications:

- *VMS System Software Handbook*
- *VMS Software Information Management Handbook*
- *VMS Software Languages and Tools Handbook*

1.16 Written Exercises

Determine the optional software supported by your system and complete the following checklist. For assistance, see your system manager, course administrator, or an experienced user of your system.

a. _____ VAX APL

b. _____ VAX BASIC

c. _____ VAX FORTRAN

d. _____ VAX C

e. _____ VAX COBOL

f. _____ VAX DIBOL

g. _____ VAX DSM

h. _____ VAX LISP

i. _____ VAX PASCAL

j. _____ VAX RPG II

k. _____ VAX BLISS-32

l. _____ VAX PL/I

m. _____ PDP-11 BASIC-PLUS-2/VAX

n. _____ VAX DATATRIEVE

o. _____ VAX DBMS

p. _____ VAX Common Data Dictionary (CDD)

q. _____ VAX Forms Management System (FMS)

r. _____ VAX ReGIS Graphics Library (RGL)

s. _____ VAX DECgraph Plotting Package

t. _____ VAX Graphics Kernel System (GKS)

u. _____ VAX DECslide Graph Presentation Utility

v. _____ VAX Terminal Data Management System (TDMS)

w. _____ ALL-IN-1 Office Menu

x. _____ VAX DECalc Spreadsheet Calculator

y. _____ VAX DECspell Verifier/Corrector

z. _____ VAX Code Management System (CMS)

aa. _____ VAX Module Management System (MMS)

bb. _____ DECnet-VAX

cc. _____ DECnet/SNA Gateway Products

dd. _____ VAX 2780/3780 Protocol Emulator

ee. _____ Any Professional Host Tool Kits

ff. _____ Other Software (Digital)

gg. _____ Other Software (Non-Digital)

1.17 Solutions to Written Exercises

To verify your work, ask your system manager, course administrator, or an experienced user of your system to check the results of your survey.

1.18 User Working Environment

When you log in, your software working environment (process) is established according to your record in the **user authorization file** (UAF). Your UAF record consists of various "fields" of information, such as user identification, working environment parameters, resource quotas, and those operations you are allowed to perform.

When you log in by entering your user name and password at your terminal, the system creates an **interactive process** with the attributes specified in your UAF record. In addition, the VMS operating system assigns you a **process name** and **process identification number** (PID), which identify you from the time you log in until you log out. When you log out, the system deletes your process. When you log in again, the system creates a new process for you.

Each piece of information about your process is called a **process parameter**. Many of the parameters associated with your process are informational. They identify you to the system and specify your terminal and auxiliary storage (disk) device. Among these are the following:

- Account name
- Default device and directory specification
- Interactive terminal specification
- Password
- Process identification number (PID)
- Process name
- User identification code (UIC)
- User name

In addition to these informational parameters, VMS associates three sets of **control parameters** with any interactive process:

- Priority
- Resource limits (Quotas)
- Privileges

Your **process priority** affects how quickly the system responds to your DCL commands. The **resource limits quotas** determine how much of a given resource (such as physical memory or processor time) you are permitted to consume. Your **process privileges** determine which protected functions of a VMS system (such as the right to control someone else's process) you are permitted to perform. Table 1–6 lists the major parameters for an **interactive process**, a process controlled by a user working from a terminal.

Your UAF record contains the values for most of these parameters; the VMS operating system defines or records the values of others when you log in. For example, your UAF record contains the value for your default interactive process priority and subprocess quota. The VMS operating system defines your process identification number (PID) and records the name of the terminal you are using when you log in. Example 1–1 displays values typically assigned to these parameters in a **nonprivileged** account, one that possesses a minimum of privileges.

Table 1–6 Parameters that Identify and Control Interactive Processes

Parameter	Function	Syntax/ Examples	Comments
Identification Parameters			
Account Name	Identifies a user whose computer usage is combined for billing purposes.	1-8 characters `GRP11`	Two or more user accounts in the user authorization file share the same account name.
Default Device and Directory Specification	Names the device and directory where, by default, you keep your files.	See Module 2 `DISK$USER:[SMITH]`	Each user typically has a unique default directory.
Interactive Terminal Specification	Names the terminal you are using.	`TXC3`	The specific terminal name depends upon the device on which you log in.
Password	Confirms your right to log in.	1-31 characters `phantom`	The system manager initially creates your password. Typically, you can change your password at any time.
Process Identification number (PID)	Identifies your process to the operating system.	8 hexadecimal digits `202002A2`	The system assigns a unique PID to each process.

Table 1–6 (Cont.) Parameters that Identify and Control Interactive Processes

Parameter	Function	Syntax/ Examples	Comments
Identification Parameters			
Process Name	Identifies your process to other users.	1-15 characters `SMITH`	Your process is usually the same as your user name. If you log in with the same user name at more than one terminal, the VMS operating system uses the terminal specification for subsequent process names.

User Identification Code (UIC)	Names the owner of a given mass storage volume (disk pack or tape reel), file, or other data structure; also used to name the owner of a given process.	1-15 alphanumeric characters, or two octal numbers separated by a comma, and enclosed by brackets. When using two octal numbers, the numbers lie between 0 and 37776, and 0 and 177776, respectively.
		`[SMITH]`
		`[11,2]`
			Your UIC is used to determine your right to access data files, or to communicate with other processes on your system. Usually each user has a unique UIC.

User Name	Names the record in the user authorization file through which you have gained system access.	1-12 characters `SMITH`	Each user name on your system is unique. Each identifies a particular UAF record.

Table 1–6 (Cont.) Parameters that Identify and Control Interactive Processes

Parameter	Function	Syntax/ Examples	Comments
Controlling Parameters			
Priority	Determines how soon operations are executed, relative to other processes present on the system.	Decimal integer between 0 and 31	Timeshare processes run at priorities from 0-15. Real-time processes have priorities from 16-32.

Privileges	Determines which protected system operations you are allowed to perform.	Refer to the *VMS DCL Concepts Manual* for a complete list of privileges.
		
		`OPER`
		`GROUP`
		`GRPNAM`
			Most users run with a minimum of privileges.

Resource Limits	Determines how much of certain system resources you can consume.	Decimal integers, the range of which depends on the parameter. See the *VMS DCL Dictionary* for a comprehensive list. Open file quota: 20	

```
$
$ SHOW PROCESS/ALL
                                  ❶                    ❷
19-APR-1989 16:56:08.99    User: SMITH  ❸ Process ID:    202002A2
                           Node: BROWNY    Process name: SMITH_1 ❹

Terminal:        LTA28:  (ZK1123/LC-4-2)
User Identifier: [ADMIN,SMITH] ❺
❻ Base priority:   4
Default file spec: DISK$USER:[SMITH]
                                                  ❼
Devices allocated: BROWNY$LTA28:

❽ Process Quotas:
 ❾ Account name: PUBLIC
  CPU limit:                        Infinite  Direct I/O limit:        30
  Buffered I/O byte count quota:       29872  Buffered I/O limit:      30
  Timer queue entry quota:                20  Open file quota:         38
  Paging file quota:                   19538  Subprocess quota:         2
  Default page fault cluster:             64  AST quota:               17
  Enqueue quota:                         600  Shared file limit:        0
  Max detached processes:                  0  Max active jobs:          0

 Accounting information:
  Buffered I/O count:        3655  Peak working set size:          737
  Direct I/O count:           381  Peak virtual size:             3348
  Page faults:               6658  Mounted volumes:                  0
  Images activated:            41
  Elapsed CPU time:       0 00:01:00.12
  Connect time:           0 08:12:40.83

❿ Process privileges:
  TMPMBX                may create temporary mailbox
  NETMBX                may create network device

 Process rights identifiers:
  INTERACTIVE
  LOCAL
  SYS$NODE_BROWNY

 Process Dynamic Memory Area
     Current Size (bytes)          25600    Current Total Size (pages)      50
     Free Space (bytes)            21984    Space in Use (bytes)          3616
     Size of Largest Block         21872    Size of Smallest Block           8
     Number of Free Blocks             4    Free Blocks LEQU 32 Bytes        1

⓫ Processes in this tree:

 Chocoholic
   SMITH_1 (*)
 $
 $ SHOW WORKING_SET      ⓬              ⓭              ⓮
   Working Set      /Limit= 1024   /Quota= 2048    /Extent= 4096
   Adjustment enabled    Authorized Quota= 2048  Authorized Extent= 4096
 $
```

Example 1–1 Displaying the Values of Your Process Parameters

Notes on Example 1–1:

Example 1–1 displays the values typically assigned to the process parameters of nonprivileged users on a VMS system. Remember that the system manager defines some parameters in the UAF record of each user; VMS defines or records others when the user logs in. Therefore, the values of some parameters are the same each time the user logs in (unless the system manager changes them). Others are different. Module 2 discusses the values in a UAF record and ways to modify them.

❶ LTA28 is the name of the interactive terminal. Generally you can work at any available terminal at your installation.

❷ SMITH is the user name of the process. Enter the user name during the login process. The system uses it to locate the proper record in the system User Authorization File (UAF), the database of authorized users of the system.

❸ 202002A2 is a hexadecimal number known as the process identification number (PID). Each process on the system is assigned a unique PID. The system uses the PID to associate process requests with the requesting process. When you refer to the PID, you can omit leading zeros.

❹ VMS assigns a unique process name to each process on the system, which it equates to a unique PID. When you are the first person to log in with a user name, the process name is set to the user name. When you are not the first person to log in with a particular user name, the system sets your process name to the name of the terminal on which you logged in.

❺ The user identification code (UIC) identifies the process access rights. The UIC consists of two numbers separated by a comma and enclosed in brackets. The system manager can equate strings of alphanumeric characters to replace each number. The system uses these strings to create more meaningful displays. In this example, the system manager has equated ADMIN,SMITH to the UIC for the SMITH account. The UIC for the SMITH account is [11,2]. The first number (11) is the group number. Users who share the same group number can share data among themselves, or affect each other's processes provided that their system manager has granted them the GROUP privilege. The second half of the code (2) is the member number. (Translating strings assigned to UICs is discussed in Module 4.)

❻ This value is the base priority of the process, which determines how rapidly the system responds to the user's requests (commands and program execution). The VMS operating system dynamically adjusts the actual process priority between 4 and 9, to promote sharing of the CPU.

❼ The system assigns a default device and directory to the process as specified in the user's UAF record. The default device name is equated to DISK$USER. The default directory name is SMITH. Devices, directories, and their naming conventions are discussed in later modules.

❽ For a complete listing of the resource limits (quotas) associated with a process and their abbreviations, refer to the *VMS DCL Dictionary*. All quotas except those controlling working set size are listed with the command SHOW PROCESS. Memory-related quotas are listed with the command SHOW WORKING_SET, shown at the bottom of Example 1–1.

❾ Resources used by the process are recorded under the account named PUBLIC. The system manager uses this account information to potentially charge users for their use of the system, or as a method to assess usage patterns and requirements by the user community.

❿ The system manager typically assigns privileges to users as they are needed. For a complete list of process privileges, refer to the *VMS DCL Dictionary*.

⓫ SMITH_1 is the second process currently in the process tree. It is a subprocess under the process named "Chocoholic." In this example, the user has set the top process name to "Chocoholic" by using the command SET PROCESS/NAME after logging in to the system.

⓬ The default maximum number of pages this process can have in memory is 1024. The system manager sets this value in the user's UAF record.

⓭ The VMS operating system can expand the process' working set size to a maximum of 4096 pages during an interactive session. The system manager also sets this value for the user in the UAF record.

⓮ This is the working set extent discussed in the VMS Overview section.

1.19 Written Exercises

Complete the following exercise before you read the next section. Example 1–2 displays the characteristics of a process that has many privileges on your system. Using the information displayed in the example, determine the value of each of the following parameters:

Answers	Parameters
	Account Name
	Default Device and Directory Specification
	Interactive Terminal Specification
	Process Identification Number
	Process Name
	User Identification Code
	User Name
	Priority
	CPU Limit
	Open File Quota
	Working Set Limit
	Working Set Quota
	Privileges (list them)

```
$
$ SHOW PROCESS/ALL
19-APR-1989 17:05:08.99   User: GABRIEL    Process ID:   20200242
                          Node: TRMPET   Process name:   Bugler

Terminal:        LTA28:  (ZK1123/LC-4-2)
User Identifier: [ULTIMATE,GABRIEL]
Base priority:   4
Default file spec: COMPACT$DISK:[GABRIEL]

Devices allocated: BROWNY$LTA28:

Process Quotas:
 Account name: MUSIC
 CPU limit:                       Infinite  Direct I/O limit:       100
 Buffered I/O byte count quota:      50000  Buffered I/O limit:     100
 Timer queue entry quota:               80  Open file quota:         50
 Paging file quota:                  49978  Subprocess quota:         8
 Default page fault cluster:            64  AST quota:              100
 Enqueue quota:                        600  Shared file limit:        0
 Max detached processes:                 0  Max active jobs:          0

Accounting information:
 Buffered I/O count:       8241  Peak working set size:      10268
 Direct I/O count:          763  Peak virtual size:          24376
 Page faults:             37681  Mounted volumes:                1
 Images activated:           35
 Elapsed CPU time:      0 00:13:37.81
 Connect time:          0 01:25:13.34

Process privileges:
 CMKRNL               may change mode to kernel
 TMPMBX               may create temporary mailbox
 OPER                 operator privilege
 NETMBX               may create network device

Process rights identifiers:
 INTERACTIVE
 LOCAL
 SYS$NODE_BROWNY

Process Dynamic Memory Area
    Current Size (bytes)          25600    Current Total Size (pages)      50
    Free Space (bytes)            21744    Space in Use (bytes)          3856
    Size of Largest Block         21712    Size of Smallest Block           8
    Number of Free Blocks             4    Free Blocks LEQU 32 Bytes        1

Processes in this tree:
```

Example 1–2 (Cont'd on next page.) Process Parameters of an Interactive Process

```
Satchmo
  Bugler (*)
$
$
$ SHOW WORKING_SET
  Working Set       /Limit= 1024   /Quota= 4096    /Extent= 8192
  Adjustment enabled    Authorized Quota= 4096  Authorized Extent= 8192
$
$
```

Example 1–2 Process Parameters of an Interactive Process

1.20 Solutions to Written Exercises

Compare your answers with those shown below. For additional information, speak with your course administrator.

Answers	Parameters
MUSIC	Account Name
DISK$COMPACT:[GABRIEL]	Default Device and Directory Specification
LTA28	Interactive Terminal Specification
20200242	Process Identification Code
Bugler	Process Name
[ULTIMATE,GABRIEL]	User Identification Code
GABRIEL	User Name
4	Priority
Infinite	CPU Limit
50	Open File Quota
1024	Working Set Limit
4096	Working Set Quota
	Privileges (list them)
CMKRNL	
TMPMBX	
OPER	
NETMBX	

1.21 Learning Aids

You should be familiar with two types of learning aids for the VMS system: the **documentation** and **course offerings**.

1.21.1 Software Documentation

There is extensive documentation on every component of the VMS system. The *Overview of VMS Documentation* and the *VMS Master Index* provide a summary and an index of the documentation set.

1.21.2 Course Offerings

The Educational Services department of Digital Equipment Corporation offers a wide variety of lecture/lab and text-based courses. For descriptions and scheduling information on currently offered courses, contact the Digital Customer Training Center nearest you, or your Digital sales representative.

MANAGING SYSTEM USERS

2.1 Introduction

The resources of a single VMS system are limited. A given configuration has a fixed amount of physical memory and disk storage. It also has a fixed number of peripheral devices. All system users must share these resources.

The VMS operating system restricts each user's access to system resources. By controlling these restrictions, the system manager can promote fair and effective resource sharing.

The VMS operating system also restricts each user's ability to affect other users and the VMS operating system itself. The system manager and individual users can maintain system security by controlling these restrictions.

The principal VMS operating system features that regulate individual processes are:

- Priority
- Privileges
- Limits
- Disk quotas
- Protection
- Access mode

To define restrictions on individual user processes and to record system activity, the system manager uses certain VMS operating system utilities and DCL commands. These utilities and commands create and maintain the following files, which are discussed in this module:

- User authorization file (SYS$SYSTEM:SYSUAF.DAT)
- Quota files on disk volumes ([000000]QUOTA.SYS)
- Login files (for example, SYS$MANAGER:SYLOGIN.COM)
- Accounting file (SYS$MANAGER:ACCOUNTNG.DAT)

The system manager or operator must also respond to requests for assistance and send messages to users. The manager uses several utilities, including the Mail and Phone utilities, to accomplish this.

2.2 Objectives

To divide limited system resources among users and processes, a system manager should be able to:

- Identify the relationships between process characteristics and the information stored in UAF records.
- Add a user account to the system by creating the necessary user authorization records, directories, and disk quotas.
- Remove a user account from the system.
- Regulate the use of system resources.
- Regulate processes running on the system.
- Restrict the abilities of certain users or applications.
- Communicate with system users and operators.

2.3 Resources

The following publications are referenced in this module:

1. *VMS DCL Dictionary*
2. *VMS System Manager's Manual*
3. *Guide to Setting Up a VMS System*
4. *Guide to Maintaining a VMS System*
5. *VMS SYSMAN Utility Manual*
6. *VMS AUTHORIZE Utility Manual*
7. *Guide to VMS System Security*

2.4 Defining the User Environment

2.4.1 The User Authorization File (UAF)

Users are identified on each VMS system by their user names. The system manager defines user names in the user authorization file (UAF). The UAF contains one record for each user who is authorized to log in to the system. Each UAF record consists of the user name, an encoded password and several other fields. The system manager can modify any field in a UAF record.

The manager uses the Authorize utility to modify the contents of UAF records. This utility enables the manager to:

- Specify who may use the system
- Define the user's default device and directory
- Limit the amount of space in memory for each user
- Permit interaction between users by establishing groups
- Run a command procedure for any user at login time
- Limit the number of files a user can have open at any one time
- Set limits on other resources

Each time you log in, the VMS operating 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. Example 2–1 shows the contents of a typical UAF record.

Some of the fields contain information that is typically common to users in the same group (see Table 2–1). Other fields contain information that is normally unique for each user (see Table 2–2).

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

❶ Username:  SMITH                          ❷ Owner:  MARY SMITH
❸ Account:  GRP11                           ❹ UIC:    [11,2] ([ADMIN,SMITH])
❺ CLI:       DCL                             ❻ Tables:  DCLTABLES
❼ Default:   DISK$USER:[SMITH]
❽ LGICMD:    SYS$MANAGER:GRP11LOGIN
❾ Login Flags:  Diswelcome Disnewmail
❿ Primary days:    Mon Tue Wed Thu Fri
  Secondary days:                       Sat  Sun
  Primary   000000000011111111112222    Secondary  000000000011111111112222
  Day Hours 012345678901234567890123    Day Hours  012345678901234567890123
  Network:    --------##########------              -----  No access  ------
  Batch:      --------##########------              -----  No access  ------
  Local:      ##### Full access ######              ##### Full access ######
  Dialup:     --------##########------              -----  No access  ------
  Remote:     --------##########------              -----  No access  ------
  Expiration:             (none)    Pwdminimum:   6   Login Fails:     0
  Pwdlifetime:         180 00:00    Pwdchange:       (pre-expired)
  Last Login:             (none) (interactive),        (none) (non-interactive)
⓫ Maxjobs:           0  Fillm:        20  Bytlm:      4096
  Maxacct jobs:      0  Shrfillm:      0  Pbytlm:        0
  Maxdetach:         0  BIOlm:         6  JTquota:    1024
  Prclm:             2  DIOlm:         6  WSdef:      1024
  Prio:              4  ASTlm:        10  WSquo:      2048
  Queprio:           4  TQElm:        10  WSextent:   4096
  CPU:         (none)  Enqlm:        10  Pgflquo:   10000
⓬ Authorized Privileges:
    GROUP TMPMBX NETMBX
  Default Privileges:
    TMPMBX NETMBX

UAF> EXIT
%UAF-I-NOMODS, no modifications made to system authorization file
%UAF-I-NAFNOMODS, no modifications made to network authorization file
%UAF-I-RDBNOMODS, no modifications made to rights database
$
$
```

Example 2–1 UAF Record

Table 2–1 Fields in a UAF Record Usually Common to Groups of Users

Field	Meaning	AUTHORIZE Qualifier	Key†
Account	Used by Accounting utility to identify processes whose activities are billed in common. To include spaces in the account name, enclose it in quotation marks.	/ACCOUNT	❸
CLI	Command language interpreter. Usually DCL.	/CLI	❺
Tables	Specifies which CLI table the process will use. (Create a new CLI table by using the Command Definition utility to copy an existing CLI table. Then use the same utility to modify the new table by removing or adding commands to it.)	/CLITABLES	❻
LGICMD	Specifies a system-wide or group-wide login command file. Commands in this file are executed before the user can issue commands interactively or through a batch command file. The manager typically assigns the same login file to all members of a group, or to all users on the system. (If this field is left blank, VMS executes the LOGIN.COM file in the user's default directory before executing user commands.)	/LGICMD	❽
Login flags	Special restrictions during and after logging in to the system.	/FLAGS	❾
Primary days Secondary days	Access allowed during specific times and days, and with specific access modes.	/ACCESS /PRIMARY /INTERACTIVE /NETWORK /BATCH /LOCAL /DIALUP /REMOTE	❿
Limits	Used to restrict use of various resources by process.	One qualifier for each limit. For example: /TQELM /ASTLM	⓫

†Refer to Example 2–1 for key numbers.

Table 2–1 (Cont.) Fields in a UAF Record Usually Common to Groups of Users

Field	Meaning	AUTHORIZE Qualifier	Key†
Privileges	Privileges to add to user record or remove from user record. Default privileges are given to user at login. Authorized privileges are given when user enters `SET PROCESS/PRIVILEGE` command.	`/PRIVILEGES` `/DEFPRIVILEGES` Qualifiers have a keyword for each privilege. For example, GRPNAM (to add this privilege) and NOGRPNAM (to delete it).	⓬

†Refer to Example 2–1 for key numbers.

Table 2–2 Fields in a UAF Record Usually Unique to Each User

Field	Meaning	AUTHORIZE Qualifier	Key†
User Name	Identifies user for whom the VMS operating system creates process. Typed in response to `Username:` prompt.	User name is a parameter, not a qualifier.	❶
Password	Confirms identity of user. Typed in response to `Password:` prompt. Note that it is possible to create an "OPEN" account by specifying a null password (`/NOPASSWORD`). The `Password:` prompt is not output for OPEN accounts.	`/PASSWORD`	N/A‡
UIC	Used to enforce protection of devices, volumes, and files. Also used to regulate ability of process to communicate with and affect other processes. (UIC can be displayed in a numerical or character string form.)	`/UIC`	❹
Owner	Records full name of user for convenience of the system manager. Not used by the VMS operating system. To allow spaces in the name, enclose it in quotation marks.	`/OWNER`	❷

†Refer to Example 2–1 for key numbers.
‡Not displayed, to preserve integrity.

Table 2–2 (Cont.) Fields in a UAF Record Usually Unique to Each User

Field	Meaning	AUTHORIZE Qualifier	Key†
Default	Used by the VMS operating system to establish both initial default device and initial default directory at login time. Note that it is preferable to identify the default device by a logical name based on the volume label (such as WORK1) rather than by a physical device name (such as DUA1).	/DEVICE /DIRECTORY	❼

†Refer to Example 2–1 for key numbers.

2.4.2 The Authorize Utility

A standard UAF file, SYS$SYSTEM:SYSUAF.DAT, is shipped with every VMS system. This file contains four Digital-supplied UAF records:

- SYSTEM
- FIELD
- SYSTEST and SYSTEST_CLIG
- DEFAULT

Table 2–3 explains the purpose of these four UAF records.

Table 2–3 Standard User Records in the User Authorization File

User Name (and Initial Password) of UAF Record	Purpose of this Record	Comments
SYSTEM (MANAGER)[1]	For software installation, system bootstrapping, and system problem diagnosis	Do not log in as SYSTEM for routine system management functions. There are too many dangerous privileges, particularly BYPASS. Create a UAF record with UIC [1,4] and SETPRV, but without dangerous privileges, to use for routine system management functions. Substitute the READALL privilege for BYPASS.
FIELD (SERVICE)[2]	For running hardware diagnostics	
SYSTEST (UETP)[2] SYSTEST_CLIG	For running the User Environment Test Package (UETP), which tests the VMS operating system hardware and software	
DEFAULT (USER)[1]	Not possible to log in as DEFAULT	The AUTHORIZE command ADD uses this record to supply default values for new UAF records.

[1] Records present whenever a UAF is created. They cannot be removed or renamed. The password DEFAULT cannot be changed. System security requires that you change the password SYSTEM.
[2] Records present in the UAF distributed with the VMS system. Removal is possible, but not recommended. System security requires that you change the passwords of SYSTEST and FIELD.

The system manager can add any number of records to the SYSUAF.DAT file by using the Authorize utility. AUTHORIZE also enables the manager to modify, copy, or remove currently existing records. However, the manager normally leaves the Digital-supplied UAF records alone (except for changing their passwords). Each time the manager adds a new record, the DEFAULT record provides values for all fields except those the manager specifies.

To invoke the Authorize utility, use the RUN command as shown in Table 2–4.

Table 2–4 Starting the Authorize Utility

Step	DCL Commands	Comments
1	Log in as SYSTEM	You must be in a system account (group number 0-10) to run the Authorize utility. To accomplish this, log in using the SYSTEM user name or another user name that specifies a system UAF record, or give yourself the SYSPRV privilege while in a nonsystem process. The SYSPRV privilege allows you to do the same things you can do in a system process.
2	`$ SET DEFAULT SYS$SYSTEM`	The Authorize utility always manages the SYSUAF.DAT file stored in the current default directory for your process. To affect the system-wide UAF file, SYS$SYSTEM:SYSUAF.DAT, set your default to SYS$SYSTEM.
3	`$ RUN AUTHORIZE`	If the SYSUAF.DAT file does not exist, AUTHORIZE asks whether you want to create a SYSUAF.DAT. If you answer "yes," it creates one. If you know SYS$SYSTEM:SYSUAF.DAT really does exist, answer "no." Then: 1. Make sure that the system logical name SYSUAF points to SYS$SYSTEM:SYSUAF.DAT, if the logical name exists. Module 5 discusses how to display and modify system logical names in more detail. 2. Follow steps 1-3 again.

The prompt for the Authorize utility is UAF>. In Figure 2–1, a UAF record for SMITH is added; SMITH is then able to log in.

For a summary of UAF commands, see Table 2–5. More detailed descriptions and examples follow in Table 2–6.

Table 2–5 Summary of AUTHORIZE Commands

Command	Function
ADD	Adds a user
MODIFY	Modifies a user record
REMOVE	Removes a user
RENAME	Renames a user record
COPY	Copies one user record to another
LIST	Creates a file of user names that can be printed
SHOW	Displays user records
HELP	Lists commands and qualifiers available
EXIT	Returns to DCL

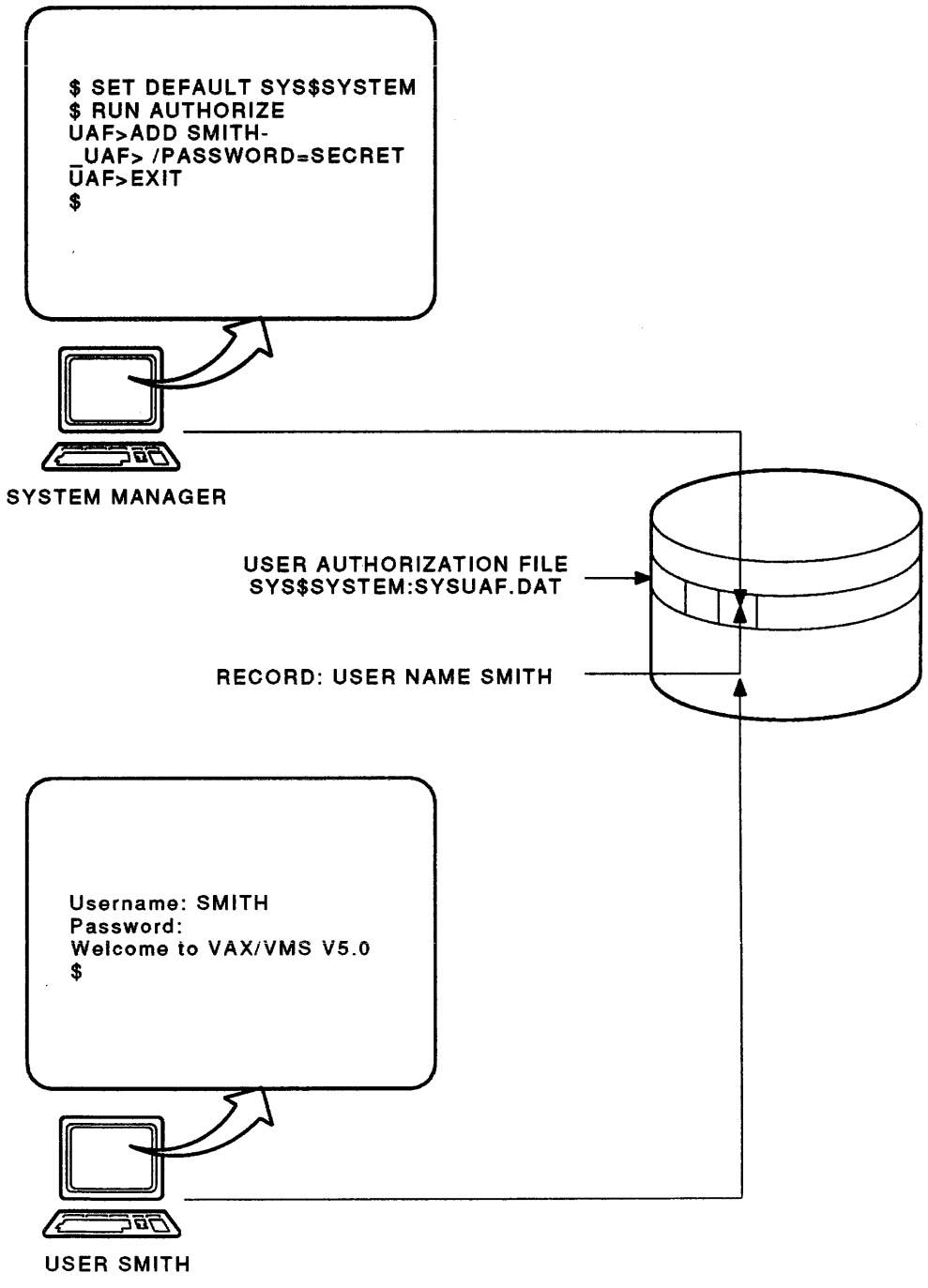


Figure 2–1 Adding and Using a UAF Record

Table 2–6 Managing the User Authorization File with the Authorize Utility

Operation on the UAF	Command Format
Displays the complete UAF record for account JONES on your terminal.	`UAF> SHOW user-name` `UAF> SHOW JONES`
Displays an abbreviated list of all accounts in the UAF file on your terminal.	`UAF> SHOW/BRIEF *`
Displays an abbreviated list on your terminal of all accounts starting with the letter "B."	`UAF> SHOW/BRIEF B*`
Writes a brief list of all accounts to the file SYSUAF.LIS.	`UAF> LIST user-name` `UAF> LIST *`
Writes a list of complete values for all accounts starting with the letter "B" to the file SYSUAF.LIS.	`UAF> LIST/FULL B*`
Adds a new user record, copying all values not specified by qualifiers from the DEFAULT record (If the /PASSWORD qualifier is omitted, the password USER is supplied.)	`UAF> ADD new-user-name [/qualifier...]` `UAF> ADD SMITH /PASSWORD=SECRET`

Adds a new user record, copying all values (except password) not specified by qualifiers from an existing user record (You must specify the password, even if it is to remain the same.)	`UAF> COPY existing-user-name new-user-name -`
	`_UAF> /PASSWORD=password [/qualifier][,...]`
	`UAF> COPY SMITH JONES /PASSWORD=FRANK`

Modifies an existing user record.	`UAF> MODIFY user-name /qualifier[,...]` `UAF> MODIFY SMITH /DEVICE=WORKDISK`
Modifies all existing user records.	`UAF> MODIFY * /qualifier[,...]` `UAF> MODIFY * /PRCLM=5`

Table 2–6 (Cont.) Managing the User Authorization File with the Authorize Utility

Operation on the UAF	Command Format

Modifies all existing user records with the same group UIC.	`UAF> MODIFY [group-number,*] -`
	`_UAF> /qualifier[,...]`
	`UAF> MODIFY [310,*] /LGICMD=GRP310:LOGIN.COM`

Renames an existing user record. (You must specify the password, even if it is to remain the same.)	`UAF> RENAME existing-user-name -`
	`_UAF> new-user-name /PASSWORD=password`
	`UAF> RENAME PRAUSS MASON /PASSWORD=JAR`

Modifies the DEFAULT record. (The qualifier /PASSWORD does not affect the default password, which is always USER.)	`UAF> MODIFY DEFAULT /qualifier[,...]`
	`UAF> MODIFY DEFAULT -`
	`_UAF> /DEV=DISK_SCRATCH /DIR=[GUEST]`

Removes an existing user record.	`UAF> REMOVE user-name` `UAF> REMOVE MASON`

Example 2–1 showed the output from a SHOW command, which displays records alphabetically in full format by default. Example 2–2 shows the output from the LIST command, which lists records alphabetically in brief format by default.

```
$
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF> LIST
%UAF-I-LSTMSG1, writing listing file
%UAF-I-LSTMSG2, listing file SYSUAF.LIS complete
UAF> EXIT
%UAF-I-NOMODS, no modifications made to system authorization file
%UAF-I-NAFNOMODS, no modifications made to network authorization file
%UAF-I-RDBNOMODS, no modifications made to rights database
$
$
$ TYPE SYSUAF.LIS

Username          UIC         Account  Privs Pri Directory

BECKER          [11,35]                Normal  4 SYS$SYSDEVICE:[BECKER]
COVERDALE       [101,6]                All     4 SYS$SYSDEVICE:[COVERDALE]
DEFAULT         [200,200]              Normal  4 Disuser
DUFFY           [101,5]                All     4 SYS$SYSDEVICE:[DUFFY]
FAL$SERVER      [376,373]   DECNET     Normal  4 Disuser
FIELD           [1,10]      FIELD      All     4 SYS$SYSROOT:[SYSMAINT]
HENDRICKS       [200,1]                All     4 SYS$SYSDEVICE:[HENDRICKS]
HOWELL          [101,4]                All     4 SYS$SYSDEVICE:[HOWELL]
LPS$SERVER      [477,477]   LPS40      Normal  4 SYS$SYSDEVICE:[LPS$SERVER]
MAIL$SERVER     [376,374]   DECNET     Normal  4 SYS$COMMON:[MAIL$SERVER]
MARSH           [11,40]                All     4 CACAO$DUA0:[MARSH]
MATTHEWS        [101,1]                All     4 SYS$SYSDEVICE:[MATTHEWS]
NOTES$SERVER    [376,377]   DECNET     Normal  4 SYS$SYSDEVICE:[NOTES$SERVER]
PHONE$SERVER    [376,372]   DECNET     Normal  4 SYS$SPECIFIC:[PHONE$SERVER]
REGNELL         [101,3]                All     4 SYS$SYSDEVICE:[REGNELL]
ROUNDS          [101,7]                All     4 CACAO$DUA0:[ROUNDS]
SPM             [1,100]     SYSTEM     All     4 SYS$SYSDEVICE:[SPM]
SYSTEM          [1,4]       SYSTEM     All     4 SYS$SYSROOT:[SYSMGR]
SYSTEST         [1,7]       SYSTEST    All     4 SYS$SYSROOT:[SYSTEST]
SYSTEST_CLIG    [1,7]       SYSTEST    All     4 Disuser
VPM$SERVER      [376,375]   DECNET     Normal  4 SYS$SPECIFIC:[VPM$SERVER]
WOODS           [101,10]               All     4 SYS$SYSDEVICE:[WOODS]
YWOSKUS         [200,2]     3F4        Normal  4 SYS$SYSDEVICE:[YWOSKUS]
$
```

Example 2–2 Brief List of UAF Records

Note that the privileges listed in Example 2–2 are categories of privileges, not actual privileges. The category of the highest privilege owned by the user is listed, even if the user does not have all privileges in that category.

2.4.3 Creating a User Account

This section illustrates the basic steps needed to add a user to the system. The amount of effort needed to add a user to your system depends on a number of circumstances, including:

- Variety of user categories: Are all users the same in terms of resource requirements and privileges, or do they widely differ?
- Security and sharing: Does your organization demand tight security enforcement for computer-based activities? Also, how are your users related in terms of data-sharing activities?
- Management of auxiliary storage: Who controls the purchase and allocation of storage devices (disks)?

These questions make it difficult to describe "typical" steps needed to add a user account to a system. However, there are certain minimal steps needed to create any user account. These steps are described in the following sections.

2.4.3.1 Preparing to Create the Account

Before you add a user to the system (by creating a new record in SYSUAF.DAT), you must determine several things to specify the proper values for the record fields:

- How should the user be categorized?

  That is, does the new user expect to have the same account parameters and privileges as another group of existing users? It is quite common to have several different "categories" of users exist on a VMS system in which each class has specifically defined values for certain UAF record fields. Of course, it is entirely acceptable to have only a single category of users on your system, in which case the new user becomes a member of that category.

  The most important consideration when categorizing a user is which UIC group to assign to the account. Also, if you use ACLs on your system, you should determine which, if any, ACL rights are to be granted to the new account. More detailed discussions of system-wide considerations of UICs and ACL identifiers are presented in Module 8.

- Where should the user's disk files reside by default?

  That is, how should the user's default device and directory be specified? Often these values are a result of placing the user in a particular category as mentioned above, but disk configuration constraints may require you to consider this situation separately.

Once you have considered these questions, you must determine a number of values for the UAF record fields, including:

1. User name and password
2. User identification code (UIC)
3. Default device and directory
4. Special resource quotas
5. Additional security parameters (ACL identifiers, privileges, and level of file protection)
6. Login command procedures to be executed

Once you have selected the proper values for any fields that differ from your default settings for the particular user category, you are ready to create the new UAF record and disk directory.

2.4.3.2 Creating the UAF Record and Default Disk Directory

A quick look at Table 2–5 shows that the ADD command is used to create a record in the UAF. You can also use the COPY command if an existing record contains most of the information needed to create the new record. However, be sure to specify the new record's password when you create the record with the COPY command, just as you would with the ADD command.

Setting disk quotas involves using the SYSMAN utility. The SYSMAN utility is particularly powerful in a VAXcluster environment and is introduced in this discussion by example, rather than by explanation.

When a user logs in, the VMS operating system uses the information in the user's UAF record to define the default device and directory. It creates a logical name for the user (SYS$LOGIN) which translates to the user's default device and directory.

However, the directory where the user's default is set does not necessarily exist. The Authorize utility does not create it automatically. You must create this directory explicitly. If it does not exist, the user will receive an error message after logging in.

To create a directory for a user in the Master File Directory (MFD) of a disk ([000000]), use the CREATE/DIRECTORY command (see Figure 2–2).

When you create a directory for a user in the MFD, it is called a user file directory (UFD). The name of this UFD typically consists of the user's last name; for example, [SMITH] or [WASHINGTON]. (The length of a directory name is limited to 39 alphanumeric characters.)

> NOTE
> If the user's default device is a volume where disk quotas are enabled, you will not be able to create a UFD for the user until you create a quota record for the user in that volume's quota file. Disk quotas are discussed later in this module.

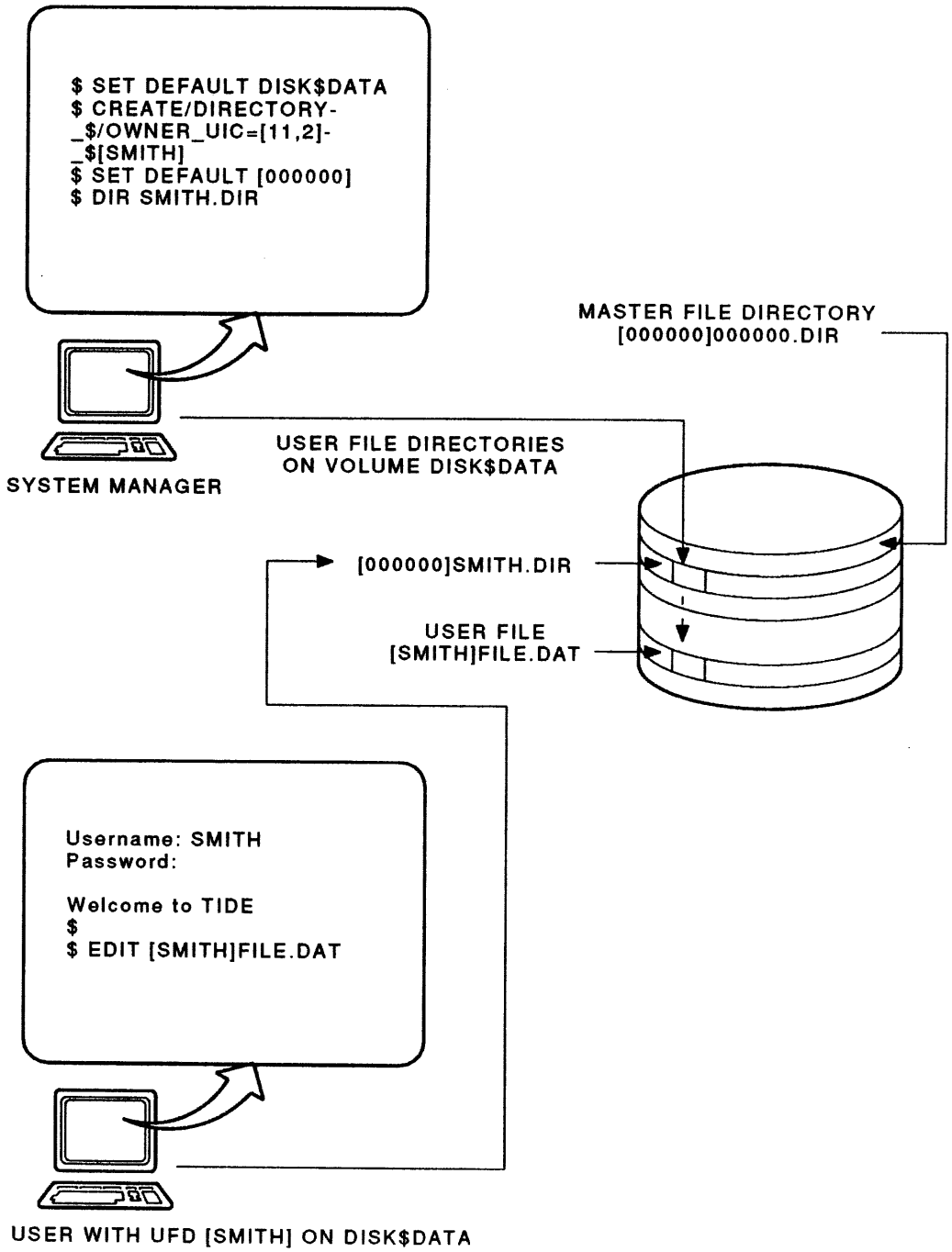


Figure 2–2 Creating a User File Directory (UFD)

Table 2–7 Creating a User's Default Directory or Other UFD

Step	Commands	Comments
1	`$ SET DEFAULT volume-name`	If the volume is not mounted, follow the instructions for mounting it in Module 4. Use the name of the device where the volume is loaded or the logical name of the volume for the volume-name part of this command. More detailed information on device names and volumes is presented in Module 4.

2	`$ CREATE/DIRECTORY -`
	`_$ /OWNER=uic ufd-name -`
	`_$ /PROTECTION=(prot-mask)`
		The UIC supplied with the /OWNER qualifier should be the user's own UIC, as recorded in the UAF. A user should always own a private UFD (one that is not shared with anyone else). The /OWNER qualifier requires SYSPRV. (See Figure 2–2.) NOTE: Users cannot create their own UFD unless they have write access to the Master File Directory (MFD) for the volume. Also, be sure to protect the new UFD with the appropriate protection mask.

3	`$ DIR volume-name:[000000]`	List the contents of the MFD to check that you did create a User File Directory in it, or set your default to [000000] on the volume and list the name of the directory, as shown in Figure 2–2.

Table 2–8 illustrates the basic steps needed to add a new user account with default values for all record fields except the following:

- Password
- UIC
- Default disk device
- Default disk directory

The steps shown in Table 2–8 illustrate only the very basic steps needed to add a user. While the above steps can be used to add a new user, you would typically want to include additional qualifiers when creating the UAF record. Which qualifiers to use - and what parameters to specify with them - are the topics for the next section.

Table 2–8 Basic Steps to Add a New User to the System

Command†	Comments
`$ SET DEFAULT SYS$SYSTEM`	Sets your default disk directory to the system directory in which SYSUAF.DAT resides.

`$ RUN SYSMAN`
`SYSMAN> DISKQUOTA ADD [11,15] -`
`_SYSMAN> /DEVICE=DISK$USER /PERMQUOTA=200 -`
`_SYSMAN> /OVERDRAFT=50`
	Invokes the System Management (SYSMAN) utility to assign the user disk quota.

`$ RUN AUTHORIZE`	Invokes the Authorize utility.

`UAF> ADD SMITH/PASSWORD=PERCHANCE -`
`_UAF> /DEVICE=DISK$USER/DIRECTORY=[SMITH] -`
`_UAF> /UIC=[ADMIN,SMITH]`
	Adds the new account SMITH, using the default UAF record values for those fields not specified in the ADD command. ADMIN and SMITH identifiers must exist for the /UIC qualifier to work.

`UAF> EXIT`	Exits from the Authorize utility and return to DCL.

`$ CREATE/DIRECTORY DISK$USER:[SMITH] -`
`_$ /OWNER=[ADMIN,SMITH] -`
`_$ /PROTECTION=(S:RWED,O:RWE,G:RE,W:E)`
	Creates the user's default disk directory on the default device as specified when the UAF record was created. Note that the disk directory can be created before the UAF record is created. The protection mask given to the new directory is only an example; the exact protection mask should reflect your particular system's security policy.

†You must be logged in to either the SYSTEM account or an account with the SYSPRV privilege enabled.

For more information on creating directory files, read the command description for CREATE/DIRECTORY in the *VMS DCL Dictionary*.

2.4.3.3 Steps for Adding a User

The following is a step-by-step summary of how to add a system user account. The example account used to illustrate these procedures is also used in a subsequent section describing the steps needed to remove the account.

1. Plan the account. As a minimum, you must decide on the user name, password, UIC, and default device and directory.
2. Log in as SYSTEM.
3. Run the Authorize utility to create a UAF record for the user.
4. 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.

   NOTE: This is an optional step. A detailed discussion of disk quotas is covered in a later section of this manual.
5. 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.)
6. Log in to the user's account using his or her user name and password to test it.

Example 2–3 reviews the steps used to add a typical system user, Mary Smith. This example assumes disk quotas are enabled on the volume containing the new user's UFD. To add a user, log in to the SYSTEM account, where the UIC is [1,4] and the default directory is SYS$SYSTEM.

```
$
$ SET DEFAULT SYS$SYSTEM
$
❶ $ RUN AUTHORIZE
UAF> ADD SMITH /PASSWORD=ENIGMA /UIC=[11,2] -
_UAF> /DEVICE=DISK$USER /DIRECTORY=[SMITH] -
_UAF> /OWNER="Mary Smith" /ACCOUNT=GRP11
user record successfully added
identifier SMITH value: [000011,000002] added to RIGHSLIST.DAT
UAF> EXIT
system authorization file modified
no modifications made to network authorization file
rights data base modified
$
❷ $ RUN SYSMAN
SYSMAN> DISKQUOTA ADD [11,2] /DEVICE=DISK$USER /PERMQUOTA=500 /OVERDRAFT=100
SYSMAN> EXIT
$
❸ $ CREATE/DIRECTORY /OWNER_UIC=[11,2] DISK$USER:[SMITH]
$LOGOUT
```

—Now try logging in to the new SMITH account—

```
❹ Username:  SMITH
Password:
             Welcome to VAX/VMS Version 5.2
$ SHOW PROCESS

19-APR-1989 16:55:20.37       User:  SMITH    Process ID:   20200263
                              Node:  BROWNY   Process name: "SMITH"

Terminal:            TTC1:
User Identifier:     [ADMIN, SMITH]
Base Priority:       4
Default file spec:   DISK$USER:[SMITH]

Devices allocated:   BROWNY$TTC1:
$
$ SHOW DEFAULT
  DISK$USER:[SMITH]
$
$ SHOW QUOTA
  User [SMITH] has 6 blocks used, 494 available, of 500
  authorized and permitted overdraft of 100 blocks on DISK$USER
$
$
```

Example 2–3 Adding a System User

Notes on Example 2–3:

❶ Creating Mary Smith's UAF record

❷ Creating Mary Smith's quota record in the quota file for the volume whose logical name is DISK$USER (Detailed information about managing disk quotas is presented in a later section of this module.)

❸ Creating Mary Smith's UFD on the volume named DISK$USER

❹ Logging in to the SMITH account to check your work

2.4.4 Modifying the DEFAULT Record

When you use the AUTHORIZE command ADD, you are essentially performing a COPY command using the DEFAULT record as the source record. That is, these two commands are identical:

```
UAF> ADD JONES
UAF> COPY DEFAULT JONES
```

The DEFAULT record is used as a template to provide parameters for record fields not specified during the ADD or COPY operation. Use the DEFAULT command to modify the DEFAULT record's field values, for example:

```
UAF> DEFAULT /DEVICE=DISK$USER /PWDEXPIRED /PRCLM=6
%-UAF-MDFYMSG, user record(s) updated
```

You can also use the MODIFY command to alter the DEFAULT record.

> NOTE
> The **only** field you are **not** allowed to change in the DEFAULT record is the **password** field. The VMS operating system is distributed with the SYSUAF.DAT file having a DEFAULT record password of USER, and this fact is widely known.

It is your responsibility as a VMS system manager to ensure that all accounts on your system are created with a suitable password and/or password expiration. The topic of password management is detailed later in this module; additional system security topics are detailed in Module 8.

By creating a DEFAULT record that is highly representative of your "typical" user account, you can save yourself considerable time when creating accounts. Even if the DEFAULT record does not possess many of the fields needed when creating a specific account, it can usually save you considerable keystrokes by allowing you to enter only those qualifiers and parameters that differ from the DEFAULT record values.

In many VMS operating system installations, the steps in Table 2–8 are completely adequate for creating most user accounts once the system manager has properly defined the DEFAULT record in SYSUAF.DAT.

For more detail on the steps to add or delete a system user, read the chapter Managing System Users, in the *VMS System Manager's Manual*.

2.4.5 Tailoring User Accounts

The preceding sections illustrated the very basic steps needed to add a new system user. Each step required to add the user was shown in its entirety, however the parameters used to create the UAF record were quite simplified. Often you need to consider and add many more values to a new user's UAF record depending on the user's special requirements.

There are many qualifiers available for use in creating and modifying UAF records with the Authorize utility. These qualifiers provide more specific control in defining the account and the processes created as a result of those definitions.

If you invoke AUTHORIZE and enter the command HELP ADD, you will probably feel overwhelmed by the many qualifiers listed. This reaction is normal, but soon you will realize that most of the qualifiers fit into the following handful of categories:

1. Identification and environment
2. Access and security
3. Quotas and resource limits
4. Privileges

Example 2–4 shows a sample listing of a UAF record and indicates the location of these field categories starting with a numbered identifier.

The following sections cover each of these categories and their associated qualifiers. While you read about each category, you will find it useful to refer to the listing of a sample UAF record in Example 2–4.

```
❶ Username:  SMITH                              Owner:  MARY SMITH
  Account:   GRP11                                UIC:  [11,2] ([ADMIN,SMITH])
  CLI:       DCL                                Tables:  DCLTABLES
  Default:   DISK$USER:[SMITH]
  LGICMD:    SYS$MANAGER:GRP11LOGIN
❷ Login Flags:  Diswelcome Disnewmail
  Primary days:    Mon Tue Wed Thu Fri
  Secondary days:                          Sat  Sun
  Primary   000000000011111111112222   Secondary  000000000011111111112222
  Day Hours 012345678901234567890123   Day Hours  012345678901234567890123
  Network:  --------##########------              -----  No access  ------
  Batch:    --------##########------              -----  No access  ------
  Local:    ##### Full access ######              ##### Full access ######
  Dialup:   --------##########------              -----  No access  ------
  Remote:   --------##########------              -----  No access  ------
  Expiration:             (none)    Pwdminimum:   6   Login Fails:     0
  Pwdlifetime:          90 00:00    Pwdchange:       (pre-expired)
  Last Login:             (none) (interactive),         (none) (non-interactive)
❸ Maxjobs:           0  Fillm:        20  Bytlm:      4096
  Maxacct jobs:      0  Shrfillm:      0  Pbytlm:        0
  Maxdetach:         0  BIOlm:         6  JTquota:    1024
  Prclm:             2  DIOlm:         6  WSdef:      1024
  Prio:              4  ASTlm:        10  WSquo:      2048
  Queprio:           4  TQElm:        10  WSextent:   4096
  CPU:          (none)  Enqlm:        10  Pgflquo:   10000
❹ Authorized Privileges:
    GROUP TMPMBX NETMBX
  Default Privileges:
    TMPMBX NETMBX
```

Example 2–4 UAF Record Field Categories

Notes on Example 2–4:

❶ Identification and environment fields

❷ Access and security fields

❸ Quotas and resource limits fields

❹ Privileges fields

2.4.5.1 Identification and Environment Fields

Certain UAF record fields contain information used by the system for accounting purposes, user identification, and initializing user's environment at login time. Table 2–9 lists the qualifiers used with the commands ADD and MODIFY to set these field values.

Table 2–9 AUTHORIZE Qualifiers for Identification and Environment Fields

Qualifier	Function
/ACCOUNT=account-name	Displays a 1 to 8 alphanumeric character string identifying the account.
/CLI=cli-name	Displays name of the default command language interpreter.
/CLITABLES=table-name	Specifies user-defined CLI tables for the account.
/DEVICE=device-name	Specifies the default disk device at login.
/DIRECTORY=directory-name	Specifies the default disk directory at login.
/LGICMD=file-spec	Specifies the name of the login command procedure to be executed after the system-wide login procedure is executed (if one exists).
/OWNER=owner-name	Specifies a 1 to 31 character string identifying the owner of the account.
/UIC=uic	Specifies the user identification code (UIC) for the account.

2.4.6 Login Command Procedures

The VMS operating system provides a powerful capability to further specify the user's environment once the user has successfully logged in. After creating the user's process upon logging in, the VMS operating system executes two login command procedures in the following order.

- The system manager's system-wide DCL login command procedure, SYS$MANAGER:SYLOGIN.COM, which customizes every DCL user's VMS operating system environment.
- The group-wide login command procedure, which customizes the user environment of each user in a group, or the user's own login command procedure, which customizes the environment for the user who is logging in.

To decide whether or not to execute the system manager's login command procedure, the VMS operating system looks for the system logical name SYS$SYLOGIN. If this logical name exists, the VMS operating system translates it to find the system-wide login command file. Typically, the logical name is translated to SYS$MANAGER:SYLOGIN.COM.

The VMS operating system then looks for an entry in the LGICMD field of the user's UAF record (see Example 2–4). If this field contains a value, the VMS operating system uses it to locate the group or user login command procedure. The field could contain a logical name that translates to the name of the procedure, or it could contain the name of the procedure itself. If the field is blank, the VMS operating system looks for the default user login command file, SYS$LOGIN:LOGIN.COM.

> NOTE
> The VMS operating system executes only one login procedure after the system login procedure; either the one specified in the LGICMD field or, if the LGICMD field is blank, the default procedure (SYS$LOGIN:LOGIN). If there is no other login file, the VMS operating system simply displays a prompt after it executes the system manager's login procedure.

Note that the VMS operating system executes login command procedures at the beginning of batch and network jobs as well as before interactive jobs. You can use the lexical function F$MODE in a login command procedure to determine the type of job that is executing. Then the procedure can branch to sequences of commands appropriate for that type of job.

For example, you could include the following line in the system manager's login file:

```
$ IF F$MODE() .EQS. "INTERACTIVE" THEN "$TYPE SYS$MANAGER:NOTICE.TXT"
```

There are many ways to use login command procedures to customize a user's VMS operating system environment. Four common uses are:

- Personal login only
- System and personal login
- System and group login
- Captive login

These command procedures are discussed in Table 2–10. An example display of the use of a turnkey account is shown in Example 2–5. Turnkey accounts are discussed in Table 2–10.

```
Username:  SPECULATE
Password:

            Welcome to SPECULATE

Speculate>  USE INVESTMENT DATA
Spec:  Consider it done.
Speculate>  COMPUTE FOR NEXT 10 YEARS
SPEC:  Please specify commodity.
Speculate>  GOLD
Spec:  $933,999,456,657.32
Speculate>
              .
              .
              .
```

(User continues to interact with the SPECULATE program)

```
              .
              .
              .
Speculate>  BYE
SPECULATE logged out at 20-OCT-1988  16:00:15.16
$
```

Example 2–5 Using a Turnkey Account

Table 2–10 Typical Login Command Procedures (DCL)

Login Command Files	System Logical Name SYS$SYLOGIN and UAF Record Field LGICMD
Function	**Definition**
Personal login only: DCL users create the file LOGIN.COM in their own login default directories to customize their own environments.	SYS$SYLOGIN undefined LGICMD undefined
System and personal login: The system manager creates the DCL command file SYS$MANAGER:SYLOGIN.COM to customize a common user environment for all users on the system. Users each create a file, LOGIN.COM, in their own login default directories to customize their own environments.	SYS$SYLOGIN defined as SYS$MANAGER:SYLOGIN LGICMD undefined
System and group login: The system manager creates the DCL command file SYS$MANAGER:SYLOGIN.COM to customize a common user environment for all system users. Group managers create a DCL command file to customize a common user environment for all users in their group.	SYS$SYLOGIN defined as SYS$MANAGER:SYLOGIN LGICMD defined as the file created for the group that the user belongs to (see Example 2–4).

Table 2–10 (Cont.) Typical Login Command Procedures (DCL)

Login Command Files	System Logical Name SYS$SYLOGIN and UAF Record Field LGICMD
Captive login: The system manager creates a file in the SYS$MANAGER directory, such as SYS$MANAGER:CAPTIVE.COM, to customize the user environment and prevent a user from changing it. The command procedure examines each DCL command the user enters to decide whether or not to allow it to be executed. It may even implement a private command language for the user. **Turnkey accounts** can also use the captive login method. Typically, more than one person uses a turnkey account. When users log in to a turnkey account, the login procedure runs a program for them and they communicate with that program. Normally, a turnkey account user never sees the DCL prompt (see Example 2–5).	SYS$SYLOGIN definition optional; affects captive and noncaptive users. LGICMD contains the name of a captive command procedure, such as SYS$MANAGER:CAPTIVE.COM, or a logical name translating to the captive command procedure name. A captive command procedure must contain a loop to prevent it from exiting, and the FLAGS field of the UAF record must specify the CAPTIVE, DISCTLY, and LOCKPWD flags. Table 2–13 discusses the FLAGS field of the UAF record further.

2.4.6.1 Access and Security Fields

Several UAF record fields are used to specify access and security constraints for the account. Access fields are used to limit the use of the account to particular times of the day, days of the week, and modes of access. Certain access fields (login flags) are also used to limit specific capabilities once the account has been logged in.

Security fields are used by the system to authenticate user requests for file access and other information requests. The most basic authentication procedure is the login procedure itself, and there are several fields that contain identification and control information about the account's passwords.

2.4.7 Access Times and Modes

You can limit an account's access to the system in three ways:

1. Time of day
2. Day of week
3. Access mode

You can combine these values to further specify the account's ability to gain access to the system.

The /PRIMEDAYS and /ACCESS qualifiers allow you to restrict access to users according to the day or hour they log in, or according to the mode used to log in.

With the /PRIMEDAYS qualifier, you define which days of the week are primary days and which are secondary. For example, the following command sets the primary days to be Monday through Friday for the user SMITH. (The other days of the week automatically become secondary days for this user.)

```
UAF> MODIFY SMITH/PRIMEDAYS=(MON,TUE,WED,THU,FRI)
```

With the /ACCESS qualifier, you define the types of access allowed and when access is allowed. The format for specifying times and days for access is:

```
/[NO]ACCESS=([PRIMARY], [n-m], [n], [,...] [SECONDARY], [n-m], [n], [,...]) )
```

For example, the following commands allow SMITH to gain access to the system in any manner from 8 am through 5 pm except during lunch on primary days, but prevents any access on secondary days:

```
UAF> MODIFY SMITH/ACCESS=(PRIMARY, 8-11, 13-16)
UAF> MODIFY SMITH/NOACCESS=SECONDARY
```

The qualifiers /ACCESS and /PRIMEDAYS are used to specify access for any type of access mode. Additional qualifiers allow you to specify which access modes are allowed for particular times and days. Table 2–11 lists the various types of access modes.

Table 2–11 Login Access Modes

Mode	Description
INTERACTIVE	Any kind of interactive login
LOCAL	Directly connected terminals (no LATs or modems)
DIALUP	Modem connections using telephone services (or network services emulating telephone services)
REMOTE	Virtual terminal connection across DECnet
BATCH	Batch jobs (noninteractive access method)
NETWORK	DECnet noninteractive network access, for example: file transfers, electronic mail to/from other nodes, etc.

For example, the following commands allow SMITH to use a local terminal during the day and a dial-up terminal during the evening and weekends.

```
UAF> MODIFY SMITH/LOCAL=(PRIMARY, 8-17, SECONDARY, 8-17)
UAF> MODIFY SMITH/DIALUP=(PRIMARY, 18-7, SECONDARY, 0-23)
```

Table 2–12 summarizes the AUTHORIZE qualifiers used to restrict access.

Table 2–12 AUTHORIZE Qualifiers for Access Fields

Qualifier	Function
/ACCESS[=(range[,...])]	Specifies the hours of access for all modes of access.
/BATCH[=(range[,...])	Specifies hours of access permitted for batch jobs.
/DIALUP[=(range[,...])†	Specifies hours of access permitted for dialup jobs.
/INTERACTIVE[=(range[,...])	Specifies hours of access permitted for interactive logins.
/LOCAL[=(range[,...])†	Specifies hours of access permitted for interactive logins initiated on local terminals.
/PRIMEDAYS=([NO]day[,...])	Specifies the primary and secondary days of the week for logins. Specify primary days as MON, TUE, WED, THU, FRI, SAT, and SUN. Specify secondary days as NOMON, NOTUE, NOWED, etc.
/NETWORK[=(range[,...])	Specifies hours of access permitted for network batch jobs.
/REMOTE[=(range[,...])†	Specifies hours of access permitted for interactive logins initiated by network remote terminals.

†These are interactive logins, so you can use the /INTERACTIVE qualifier to specify all three interactive access methods.

2.4.8 Login Flags

One field of the UAF record (login flags) is used to signify special user restrictions during or after the login operation. To set one or more login flags for an account, use the following format with either the ADD or MODIFY commands:

```
UAF> command username /FLAG=(flag,...)
UAF> MODIFY SMITH /FLAG=DISUSER
UAF> ADD JONES /FLAG=(DISCTLY,DEFCLI)
```

Table 2–13 describes the available flags and their purposes.

Table 2–13 Login Flag Parameters

/FLAG Parameter†	Function
AUDIT	Audits all security-relevant actions
AUTOLOGIN	Restricts this account to autologins only
CAPTIVE	Prevents user from changing any defaults at login
DEFCLI	Prevents user from changing default CLI or CLI table
DISCTLY	Disables CTRL/Y interrupts
DISFORCE_PWD_CHANGE	Disables forced user expired password changes
DISMAIL	Prevents mail delivery to this user
DISNEWMAIL	Suppresses "New Mail..." announcements
DISRECONNECT	Disables automated reconnections
DISREPORT	Disables time of last login and other security reports
DISUSER	Disables this account completely
DISWELCOME	Suppresses "Welcome to..." login message
GENPWD	Requires user to use generated passwords
LOCKPWD	Prevents user from changing password
PWD_EXPIRED	Marks password as expired
PWD2_EXPIRED	Marks second password as expired

†Any flag can be prefixed with NO to turn off the flag's intended purpose, for example: /NOLOCKPWD

One common application, a **captive login**, uses one or more of these flags to maintain tight control over the account while logged in to the system (see Table 2–10). When you add a user record to the UAF with a captive login, specify the qualifier /FLAGS=(CAPTIVE,DISCTLY,LOCKPWD), as explained in Table 2–10.

Another good security technique is to keep certain privileged user names disabled when they are not expected to be used. For example, you need the SYSTEST and FIELD accounts only occasionally. You can improve the security of these accounts by modifying their UAF records using the qualifier /FLAGS=DISUSER. As long as you set this flag, no one can log in with these user names.

When you need to log in with one of these user names, modify its UAF record again by using the qualifier /FLAGS=NODISUSER. This technique has the added benefit of retaining the account's password while the account is disabled. For example, if your Digital Field Service representative is required to access your system, you can simply enable the account /FLAG=NODISUSER and inform the representative to enter the previously used password.

2.4.9 Security Fields

As you can see from Table 2–13, there are a number of login flags that directly pertain to security, especially with password management. There are other fields within the UAF record that contain information used to control various aspects of password administration and information about the account's rights identifiers. Table 2–14 lists the qualifiers used to set these fields, as well as control the rights database.

The subject of system security and integrity is detailed in Module 8. For now, you should just become familiar with the various qualifiers and their purposes.

Table 2–14 AUTHORIZE Qualifiers for Security Fields

Qualifier	Function
/ADD_IDENTIFIER	Adds identifiers for the user name and account name to the rights database.
/EXPIRATION=time	Shows expiration date and time of the account.
/GENERATE_PASSWORD[=keyword]	Invokes the password generator to generate user passwords. Details of possible keywords are discussed in Module 8.
/MODIFY_IDENTIFIERS	Specifies whether the identifier associated with a user record is to be modified in the rights database.
/PASSWORD=(pwd1[,pwd2])	Specifies the primary and optional secondary passwords.
/PWDEXPIRED	Specifies whether a password is valid only for the first login.
/PWDLIFETIME=time	Specifies the length of time a password is valid, entered as a delta-time value.
/PWDMINIMUM=value	Specifies the minimum number of characters allowed for a password.
/REMOVE_IDENTIFIER	Specifies whether the user name and account name identifiers should be removed from the rights database when the UAF record is removed from SYSUAF.DAT. Works only for the REMOVE command.

2.4.9.1 Quotas and Resource Limits

Module 1 showed that there are many quotas and resource limits associated with a process. When a user logs in to an account, the created process is given the quotas and other limits defined in the account's UAF record. Table 2–15 lists the qualifiers used to set these and other process-related parameters.

Usually, the VMS system manager would first set the typical values for these parameters in the DEFAULT record to keep from having to input the values for each new account.

Table 2–15 AUTHORIZE Qualifiers for Quota Fields

Qualifier	Definition
/ASTLM=value	Number of ASTs the user can have queued at any one time.
/BIOLM=value	Maximum number of buffered I/O operations the user can have outstanding at any one time.
/BYTLM=value	Maximum number of bytes of nonpaged system dynamic memory that the user's job may consume at any one time.
/CPUTIME=time	The maximum CPU time a user's process can take per session, specified as a delta-time value.
/DIOLM=value	Maximum number of direct I/O operations (usually disk) that the user can have outstanding at any one time.
/ENQLM=value	Maximum number of locks that can be queued at any one time.
/FILLM=value	Maximum number of files that can be open at one time.
/JTQUOTA=value	The initial maximum number of bytes with which the job-wide logical name table is to be created.
/MAXACCTJOBS=value	Maximum number of batch, interactive, and detached processes that may be active at any one time for all users of the account. The default value of 0 represents an unlimited number.
/MAXDETACH=value	Maximum number of detached processes allowed at any one time.
/MAXJOBS=value	Maximum number of batch, interactive, detached, and network processes that may be active at any one time.
/PGFLQUOTA=value	Maximum number of pages the user's process can use in the system paging files.
/PRCLM=value	Maximum number of subprocesses that can exist at once for the user's process.
/PRIORITY=value	The default base priority for all processes created by the user.

Table 2–15 (Cont.) AUTHORIZE Qualifiers for Quota Fields

Qualifier	Definition
/SHRFILLM=value	Maximum number of shared files the user can have open at any one time.
/TQELM=value	Total number of entries in the timer queue, plus the number of temporary common event flag clusters the user can have at any one time.
/WSDEFAULT=value	The number of pages in the user's default working set.
/WSEXTENT=value	The number of pages in the user's working set extent.
/WSQUOTA=value	The number of pages in the user's working set quota.

2.4.9.2 Privileges

The VMS operating system provides a comprehensive set of privileges to allow precise access and control of various resources and operating system facilities. Normally the system manager grants only the minimum number of privileges to a typical system user. A typical system user is commonly called a **nonprivileged** user, which means the user has only these minimal privileges (as opposed to no privileges at all). The minimum set of required privileges for nonprivileged users can vary from system to system, but normally consists of the following:

- TMPMBX
- NETMBX (useful only on systems with DECnet services)

Table 2–16 lists the qualifiers used to set the account's privileges.

Table 2–16 AUTHORIZE Qualifiers for Privilege Fields

Qualifier†	Function
/DEFPRIVILEGES=([NO]privname[,...])	Specifies the list of privileges that are enabled at login time. The keyword [NO]ALL disables or enables all user privileges.
/PRIVILEGES=([NO]privname[,...])	Specifies the list of privileges granted (but not enabled) at login time.

†Any privilege keyword used with either qualifier may be prefixed with **NO** to turn off the privilege.

There are two sets of privileges associated with each UAF record. One set of privileges is specified with the /DEFPRIVILEGES qualifier; these are called the **default privileges** and are enabled once the account has successfully logged in. The other set is specified with the /PRIVILEGES qualifier; these are called the **authorized privileges**. The authorized privileges can only be enabled by the user through the explicit use of the command SET PROCESS/PRIVILEGE.

The reason for having two separate sets of privileges is to avoid accidental misuse of privileges. Sensitive privileges should never be placed in the default privilege set (/DEFPRIVILEGES), but rather in the second privilege set (/PRIVILEGES). The complete set of VMS operating system privileges is listed in Table 2–17. Detailed definitions of these privileges can be found in the *Guide to Setting Up a VMS System*.

Table 2–17 The VMS Operating System Privileges

Privilege	Function
ACNT	Suppresses accounting message
ALLSPOOL	Allocates spooled device
ALTPRI	Sets any priority value
BUGCHK	Makes bug check log entries
BYPASS	Bypasses UIC checking

Table 2–17 (Cont.) The VMS Operating System Privileges

Privilege	Function
CMEXEC	Changes mode to exec
CMKRNL	Changes mode to kernel
DETACH	Creates detached processes
DIAGNOSE	Diagnoses devices
EXQUOTA	Exceeds quota
GROUP	Affects other processes in same group
GRPNAM	Inserts in group logical name table
GRPPRV	Allows group access by means of system protection
LOG_IO	Does logical I/O
MOUNT	Executes mount ACP function
NETMBX	Creates network device
OPER	Operator privilege
PFNMAP	Maps to specific physical pages
PHY_IO	Does physical I/O
PRMCEB	Creates permanent common event clusters
PRMGBL	Creates permanent global sections
PRMMBX	Creates permanent mailbox
PSWAPM	Changes process swap mode
READALL	Reads anything as the owner
SECURITY	Performs security functions
SETPRV	Sets any privilege bit

Table 2–17 (Cont.) The VMS Operating System Privileges

Privilege	Function
SHARE	Assigns channels to nonshared device
SHMEM	Creates/deletes objects in shared memory
SYSGBL	Creates system-wide global sections
SYSLCK	Locks system-wide resources
SYSNAM	Inserts in system logical name table
SYSPRV	Accesses objects by means of system protection
TMPMBX	Creates temporary mailbox
VOLPRO	Overrides volume protection
WORLD	Affects other processes in the world

2.4.10 Removing a User Account

The exact procedure for removing a system user can vary widely from site to site, depending on the site's security policies. However, all procedures usually have these steps in common:

1. Remove the UAF record
2. Dispose of any remaining disk files
3. Remove the user's entries in volume quota files (if any)
4. Remove associated VMS operating system mail information for the account

Each step is described in more detail below. Note that all steps require you to have the SYSPRV privilege enabled.

2.4.10.1 Removing the Account

Remove an account from the system by using the REMOVE command in the Authorize utility to delete the UAF record. However, sometimes it is more convenient to first disable the record before actually removing it.

For example, there are times when you are told to immediately take action to keep a user from accessing the system (for one or more reasons), but you are not told what to do with any residual disk files that might exist. If you simply disable the account (MODIFY /FLAG=DISUSER), you have provided maximum security against access of the account, giving you time to figure out what to do with the disk files. Later, after the disk files have been properly disposed, you can go back into AUTHORIZE and remove the UAF record.

2.4.10.2 Removing the Disk Files

This is another procedure that can vary widely from site to site. In some cases, you may simply delete all files in the user's default directory tree. In other cases, again depending on the site's security and data management policies, you might have to back up the files to tape or other storage device, such as an archive disk.

Example 2–6 lists a command procedure that can be used to properly delete all files and subdirectories in a user's default directory tree.

2.4.11 Removing the Disk Quota Record

If disk quotas are enabled on the user's default volume, you should remove the user's record from that volume's quota file. The following commands summarize this procedure:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN SYSMAN
SYSMAN> DISKQUOTA REMOVE uic /DEVICE=volume-name
SYSMAN> EXIT
$
```

"Volume-name" is the name of the account's default device specified in the UAF record (/DEVICE) qualifier). The "UIC" argument is the account's UIC specified in the UAF record (/UIC) qualifier).

> NOTE
> If the account shares a UIC with other accounts, do **not** remove the quota record, since the other accounts would no longer be able to create files on that volume.

Volume quotas are detailed later in this module.

2.4.12 Removing the VMS Operating System Mail Information

You remove the account's VMS mail information using the following commands:

```
$ MAIL
MAIL> REMOVE username
MAIL> EXIT
$
```

```
$! DELTREE.COM (P1 = name of device, P2 = name of directory)
$!
$!   Command procedure to delete a UFD and all its subdirectories.
$!   Procedure deletes files from bottom of specified directory
$!   structure to top of structure.
$!   Requires SYSPRV privilege or a system UIC
$!   Back up all useful files before running this procedure.
$!
$    SET NOON
$!
$!   Get name of directory structure to delete if not already known
$!
$   IF P1 .EQS. -- THEN INQUIRE P1 "Device, omit colon"
$   IF P2 .EQS. -- THEN INQUIRE P2 "Directory (UFD, omit brackets)"
$!
$!    Set protection to allow deletion of all files in structure
$!
$   SET PROTECTION=(SY:RWED)  'P1:['P2...]*.*;*
$!
$!    Set up counter and loop label
$!
$   COUNTER=8
$LOOP:
$!
$!   Delete files
$!
$   DELETE/LOG 'P1:['P2...]*.*;*
$!
$!   If more files, delete them also
$!
$   COUNTER=COUNTER-1
$   IF COUNTER .GT. 0 THEN GOTO LOOP
$!
$!     Delete UFD from MFD
$!
$   SET PROTECTION=(SY:RWED) 'P1:[000000]'P2.DIR
$   DELETE/LOG 'P1:[000000]'P2.DIR;1
$
```

Example 2–6 Deleting a UFD and its Subdirectories

2.4.12.1 Steps for Removing a System User

Example 2–7 shows the steps in removing system user Mary Smith, the same account used in Example 2–3 to add a system user.

To remove a user, log in as SYSTEM, or be sure your process has the SYSPRV privilege enabled. The following comments are keyed to Example 2–7.

1. Remove Mary Smith's UAF record. (NOTE: If Mary is just on a long vacation, you should instead disable her account by entering the MODIFY command with the qualifier /FLAGS=DISUSER, rather than remove it using these instructions.)
2. Remove Mary Smith's UFD and subdirectory files on her default volume, using the command procedure DELTREE.COM shown in Example 2–6. (Before this step, you should examine her files and back up those you want to save, or transfer them to another user's directory.)
3. Remove Mary Smith's record in her default volume's quota file.

   NOTE
   If Mary has quota records in quota files on other volumes, you should also remove those records. However, do not remove any quota records if Mary's UIC is not unique; if you remove the record, then other users of that UIC will not be able to use space on that volume.

4. Remove Mary Smith's VMS mail information.

```
$
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF> REMOVE SMITH
record removed from SYSUAF.DAT
identifier SMITH value: [000011,000002] removed from RIGHTSLIST.DAT
UAF> EXIT
system authorization file modified
no modification made to network authorization file
rights data base modified
$
$ @DELTREE DISK$USER SMITH
$ RUN SYSMAN
SYSMAN> DISKQUOTA REMOVE [11,2] /DEVICE=DISK$USER
SYSMAN> EXIT
$ MAIL
MAIL> REMOVE SMITH
MAIL> EXIT
$
```

Example 2–7 Removing a System User

For more information on the Authorize utility, including a description of all command qualifiers and their corresponding fields in the UAF record, read the *VMS Authorize Utility Manual.*

2.5 Laboratory Exercises

You need write access to the MFD of the class disk and to the system authorization files to do this lab.

1. Create an account called PAYROLL. The group number assigned to workers in Payroll is 322. Choose any member number from 60 - 377 (octal) that is not in use. Also create a directory and a disk quota entry for the account.

   Log in to the new account and create a small file in its directory to verify your work.

2. This account is only allowed to run a data entry program ENTRY.EXE. The name of the command procedure used to run the program is DATA.COM. These files are found in a directory pointed to by the logical name COURSE$V5SYSMGT.

   Modify the UAF record so that DATA.COM runs automatically when you log in to the PAYROLL account, and so that you cannot reach the DCL prompt.

3. Log in as PAYROLL. The DATA.COM procedure should execute automatically. The password for the procedure is GO. The procedure executes ENTRY.EXE. When ENTRY.EXE requests input, type in three numbers separated by commas.

   To test your work, log out and log in as SMITH several times. Each time, try to cause the DCL prompt to appear. (Suggestions: enter CTRL/Y, enter incorrect data at various points, enter the wrong password, etc.)

4. Do whatever is necessary to remove the account PAYROLL from the system.

2.6 Solutions to Laboratory Exercises

1. To add an account, run the AUTHORIZE and DISKQUOTA utilities and create a UFD as shown below:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF>SHOW/BR [322,*]
(Output shows 63 has not been used as a member number yet)
UAF>ADD PAYROLL /PASSWORD=JOE-
_/UIC=[322,063]-
_/DEVICE=CLASS_DISK-
_/DIRECTORY=[PAYROLL]
UAF>EXIT
$
$ SET DEFAULT CLASS_DISK:[000000]
$
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA ADD PAYROLL /DEVICE=CLASS_DISK
SYSMAN> EXIT
$
$ CREATE/DIRECTORY/OWNER=PAYROLL [PAYROLL]
$
```

   You should have been able to create a test file successfully. If not, look at these solutions carefully, set up the account properly, and try again.

   Check the owner UIC of the directory if you receive protection errors. Check the contents of the QUOTA.SYS file on CLASS_DISK if you receive quota errors.

2. To modify the account so it can only run the DATA.COM procedure, use the Authorize utility to make the account captive.

```
UAF>MODIFY PAYROLL /LGICMD=CLASS_DISK:[PAYROLL]DATA.COM-
_/FLAGS=(CAPTIVE,LOCKPWD,DISCTLY)
```

3. If you enter incorrect answers to the DATA.COM procedure or the ENTRY program, you are logged out. If you enter letters as data to ENTRY, instead of numbers, you are logged out. If you enter a CTRL/Y key sequence, you are logged out.

4. To remove the account completely, you must delete all subdirectories and files in [PAYROLL], remove CLASS_DISK:PAYROLL.DIR, remove the PAYROLL entry from the quota file, and remove the UAF record from the SYSUAF.DAT file.

2.7 Managing Users on a Daily Basis

The following sections describe the most common activities you will encounter in managing system users, including:

- Managing user processes
- Managing disk space
- Communicating with users

Certain operations tasks, such as disk backups and restorations, are covered in other modules in this course. Table 2–18 provides an overview of the methods the VMS operating system provides the system manager to regulate processes.

Table 2–18 Regulating the VMS Operating System Processes

Method of Regulation	Comments
Access to CPU	
Priority	The VMS operating system uses process priority to schedule process execution. Priority ranges from 0 (lowest) to 31 (highest). Timesharing processes have priorities 0 - 15. The VMS operating system can automatically adjust the priorities of timesharing processes to promote greater CPU sharing. Real-time processes have priorities 16 - 31. The VMS operating system does not automatically adjust the priorities of real-time processes.
Ability to affect the VMS operating system and system users	
Privileges	In many cases, exercising a privilege results in the use of physical memory or other system resources. There is no restriction on the number of times a process can exercise a privilege.

Table 2–18 (Cont.) Regulating the VMS Operating System Processes

Method of Regulation	Comments
Ability to use or not use certain DCL commands	
CLI tables	If a command is listed in the default DCL command table for the system, SYS$LIBRARY:DCLTABLES.EXE, all users on the system can use the command. However, you can create additional CLI tables and assign them to particular users to broaden or restrict their use of DCL commands.
Ability to use certain system resources	
Limits	Limits restrict the amount of a system resource a process can use at a given time. Limits are expressed numerically. Limits are also called quotas in some documentation. Do not confuse limits with disk quotas, described below.
Ability to store files on a disk volume	
Disk quotas	Disk quotas restrict the number of blocks of storage a process can use on a volume. Disk quotas are established on volumes (which can be moved from device to device) rather than on devices. Disk quotas do not have to be established for every volume on a system; you can select which volumes have this restriction.

Table 2–18 (Cont.) Regulating the VMS Operating System Processes

Access to devices, volumes, files, and interprocess communications	
Protection	Every process has a UIC. Devices, volumes, files, and interprocess communication structures have an owner UIC and a protection code. One type of the VMS operating system protection compares the process UIC with the owner UIC and protection code of the protected structure. Another type of the VMS operating system protection compares a user's access rights list with the access control list that belongs to the protected structure.
Ability to modify or delete certain data structures, such as logical names	
Access mode	Data structures created at more powerful access modes can be used at less powerful access modes; however, they cannot be modified or deleted at less powerful access modes.

2.7.1 Restricting User Activity Using DCL Commands

A process can always control itself and its subprocesses. Two typical cases are:

1. A process has created several subprocesses to do work.

   The parent process comes to a time-critical part of its program. Since all of the subprocesses and the parent process are part of the same job, they share many resources. The parent process might suspend all subprocesses prior to executing the time-critical (or resource-demanding) part of its program. After it completes this part, the parent process resumes the subprocesses.

2. A process creates a subprocess to do work.

   The program executing in the subprocess does not finish in a given amount of time. Therefore, the parent process may decide the program is in a loop and may delete the subprocess. Other processes in the job can then share the resources returned by the deleted subprocess.

Table 2–19 shows how to restrict VMS processes by:

- Changing their execution priority
- Suspending their execution
- Resuming their execution
- Stopping execution and deleting the process

You must have the GROUP privilege to control other processes in the same UIC group. If the group manager has the GROUP privilege, and decides that certain processes use too many resources, or that they are unauthorized or that they are in a loop, the manager may suspend or delete them.

The user with WORLD privilege has the same choices as the group manager, but WORLD privilege allows the user to affect any process on the system. Typically, only the system manager has WORLD privilege.

Note that, for the commands in Table 2–19 to take effect, the process being controlled must be executing. Therefore, delays may occur in affecting low priority processes on heavily loaded systems (because they are scheduled to execute only when no higher priority processes are ready.)

Table 2–19 Controlling Processes

Operation	Command Format and Examples	Comments

Suspending a process	`$ SET PROCESS/SUSPEND -`
	`_$ [/ID=pid] [process-name]`
	`$ SET PROCESS/SUSPEND/ID=21A`
	`$ SET PROCESS/SUSPEND JONES`
		Every process has a unique process identification (PID), a hexadecimal number. To refer to a process by PID, as shown in the example, use the /IDENTIFICATION qualifier. The command SHOW SYSTEM displays PIDs. You are not required to enter leading zeros.

Resuming a suspended process	`$ SET PROCESS/RESUME -`
	`_$ [/ID=pid] [process-name]`
	`$ SET PROCESS/RESUME/ID=21A`
	`$ SET PROCESS/RESUME JONES`	

Changing the base priority of a process	`$ SET PROCESS/PRIORITY=n -`
	`_$ [/ID=pid] [process-name]`
	`$ SET PROCESS/PRIORITY /ID=21A`
		Every process has a process name, unique within its UIC group. To refer to a process in your UIC group by process name, supply the name as the command parameter. To refer to a process not in your UIC group, you must use its PID, not its process name. ($SET PROCESS/PRIORITY requires ALTPRI privilege.)

Stopping and deleting a process	`$ STOP [/ID=pid] [process-name]` `$ STOP/ID=21A`	

For more information on VMS control of system resources, consult the *Guide to Maintaining a VMS System.*

2.8 Laboratory Exercises

To practice controlling user processes, log in at two terminals using two accounts and do the following exercises. Your course administrator will supply you with a second account.

1. Designate one terminal as the Manager terminal, and the other as the User terminal. Log in at both terminals. The process on the Manager terminal must have the OPER and ALTPRI privileges.
2. The User should begin entering DCL commands that produce output, such as SHOW PROCESS and SHOW SYSTEM. (You can also create a command procedure to do this repeatedly.)
3. Find out the process ID number of the User process from the Manager terminal.
4. Suspend the User process from the Manager terminal (preferably while output is appearing on the User terminal).
5. Observe the User terminal. Notice the lack of response from the keyboard (if you press any key or combination of keys, the state of the User process does not change).
6. Allow the User process to continue.
7. Lower the priority of the User process to three from the Manager terminal. The User should continue to enter DCL commands and observe the response for several minutes.
8. Raise the priority of the User process to four from the Manager terminal. Continue to observe the response to various DCL commands.
9. Stop the User process from the Manager terminal. Can the Manager allow the process to continue?

2.9 Solutions to Laboratory Exercises

1. No solution needed.
2. A command procedure you might use is:

```
$LOOP:
$ SHOW PROCESS
$ SHOW SYSTEM
$ GOTO LOOP
```

   This command procedure produces an endless loop.

3. The commands SHOW SYSTEM and SHOW USERS display the PIDs of the users currently on the system. The PID is a hexadecimal number.
4. `$ SET PROCESS /SUSPEND /ID=21400FA1`

   Substitute the appropriate PID for 21400FA1.

5. No solution needed.
6. `$ SET PROCESS /RESUME /ID=21400FA1`

   Substitute the appropriate PID for 21400FA1.

7. `$ SET PROCESS /PRIORITY=3 /ID=21400FA1`

   If the process is in your UIC group, you need only specify the process name as a parameter instead of using the /ID qualifier. For example, if the process name is SMITH, and the UIC of the process has the same group number as yours, you could enter:

```
$ SET PROCESS/PRIORITY=3 SMITH
```

   Because the priority of the process is now lower than most of the priorities of the other processes on the system, you should observe a delay in the execution of the commands entered from the User terminal.

8. `$ SET PROCESS/PRIORITY=4/ID=21400FA1`

   The response to the DCL commands entered from the User terminal should be quicker.

9. `$ STOP/ID=21400FA1`

   If the process is in your UIC group, you can omit the /ID qualifier and supply the process name as a parameter. The process cannot be continued, since the STOP command deleted the process.

2.9.1 Restricting the Use of Disk Space

UAF records contain limits on the use of many system resources. However, these records do not contain restrictions on space available to users on their default disk or on any other device.

You can control space utilization on a disk volume with the SYSMAN utility. Using this utility, you limit each user to a certain number of blocks on a volume. The user must work within this limit on that volume, remembering to purge and delete files. This limit, or quota, allows more efficient use of the disk as a storage medium.

The manager controls disk usage on a volume-by-volume basis. A choice must be made for each volume - whether to establish quotas for that volume or not. If the manager does not establish quotas on a volume, users with read and write access to the volume can use as much space on the volume as they need.

When the manager does establish quotas on a volume, VMS records the usage of space on a UIC basis. Therefore, the manager must authorize each UIC that will use space on the volume. Only authorized users can create or modify files on the volume.

2.9.1.1 The SYSMAN Utility and DISKQUOTA Functions

As of VMS Version 5, all disk quota operations are performed within the new SYSMAN utility. Within SYSMAN are a number of DISKQUOTA commands (see Table 2–20) used to manage quota files and entries.

The SYSMAN utility serves many purposes for the VMS system manager. The primary importance of SYSMAN is that it allows a system manager to conveniently manage system resources on a cluster-wide basis from a single VMS system. Since the topic of VAXclusters is not within the scope of this course, capabilities other than DISKQUOTA commands will not be discussed. A complete description of the SYSMAN utility can be found in the *VMS SYSMAN Utility Manual*.

To run the SYSMAN utility, make sure your process has the OPER privilege enabled, then use the following command:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN>
```

To obtain help on the various DISKQUOTA commands within SYSMAN, use the command DISKQUOTA HELP within the SYSMAN utility.

Table 2–20 DISKQUOTA Commands Within the SYSMAN Utility

Command Format†	Function
`DISKQUOTA ADD uic`	Adds an entry to a disk quota file and initializes its usage count to zero.
`DISKQUOTA CREATE`	Creates and enables a quota file for a disk volume that does not currently contain one. It is strongly recommended that you do **NOT** create and enforce quotas on the system disk.
`DISKQUOTA DISABLE`	Suspends the maintenance and enforcement of disk quotas on a volume.
`DISKQUOTA ENABLE`	Resumes quota enforcement on a disk volume containing an existing quota file.
`DISKQUOTA MODIFY uic`	Changes an entry in a quota file, or adjusts default values for quotas and overdrafts. Default values for entries on the volume are taken from the entry [0,0].
`DISKQUOTA REBUILD`	Updates a quota file, adding new UICs and correcting usage counts for each user on the volume.
`DISKQUOTA REMOVE uic`	Deletes an entry from the quota file.
`DISKQUOTA SHOW`	Displays quotas, overdrafts, and usage counts.
`EXIT`	Exits user from the SYSMAN utility.
`HELP DISKQUOTA`	Obtains help information on the DISKQUOTA commands.

†For all of these commands, be sure to use the qualifier **/DEVICE=volume-name** to specify the proper volume; otherwise SYSMAN will use either the last volume specified with the most recent **/DEVICE=volume-name**, or the volume associated with your current default directory (if you had not yet used **/DEVICE=volume-name** in a DISKQUOTA command. Most of these commands require SYSPRV.)

2.9.2 Establishing Quotas on a Volume

To establish quotas on a volume, you must create a quota file for the volume in the volume's MFD (directory [000000]). You can create only one quota file on any given volume. The name of the quota file is `volume-name:[000000]QUOTA.SYS`, for example: `DISK$USER:[000000]QUOTA.SYS`.

The exact steps for creating a quota file on a volume depend on whether the volume is new (just created), or if it already exists and has UFDs and files owned by one or more UICs.

2.9.3 Establishing Quotas on a New Volume

Table 2–21 lists the sequence of commands used to create a quota file for a newly created volume called DISK$DATA. (The commands needed to create a new disk volume are discussed in Module 4.)

Table 2–21 Establishing Quotas on a New Volume Called DISK$DATA

Steps	Commands	Comments
1	Log in as SYSTEM	You can alternatively give your current process the OPER privilege to use the SYSMAN utility.
2	`$ RUN SYS$SYSTEM:SYSMAN`	Invoke the SYSMAN utility.
3	`SYSMAN> DISKQUOTA CREATE -` `_SYSMAN> /DEVICE=DISK$DATA`	Creates the file DISK$DATA:[000000]QUOTA.SYS and automatically enables quotas on the volume.

After you create the quota file for a volume, add one entry for each UIC you want to authorize for that volume. Use the SYSMAN utility to add each entry. Each entry contains the following fields:

- UIC
- Usage
- Permanent Quota
- Overdraft

You must enter a value for the UIC field. You may enter values for the Permanent Quota and Overdraft fields or allow the SYSMAN utility to supply a default value, which is taken from the entry [0,0]. Entry [0,0] is automatically created when the quota file is created with the command DISKQUOTA CREATE. The SYSMAN utility supplies a value of 0 as the initial value for the Usage field. VMS updates this field while the user is working to reflect the amount of space the user's UIC owns on the volume.

Table 2–22 explains each of these fields further. Example 2–8 lists several entries from a sample quota file. SYSMAN displays quota entries in numerical order according to the value in the UIC field of each entry. You may not be able to see that the listing is in numerical order, however, because the utility displays some UIC values using their identifier equivalents. For example, the UIC value [1,4] is typically associated with the identifier name SYSTEM. As a result, the SYSMAN utility displays the name SYSTEM as the value in the UIC field instead of the numerical UIC value (see Example 2–8).

Table 2–22 Fields in a Quota File Record

Field	Function	DISKQUOTA Qualifier
UIC	Identifies the user who is permitted to use the volume. Note that files are owned by UICs, not by user names. Therefore, if more than one user shares the same UIC, all of them have the same access to files. They also share the quota assigned to that UIC for the volume. When you log in, the VMS operating system reads your UAF record to determine your UIC.	Specify the UIC as a parameter, not as a qualifier, in DISKQUOTA commands.
Usage	Shows the number of blocks of storage this UIC owns.	None. This value is updated by the VMS operating system as files are created by the UIC. It is not assigned by the system manager.
Permanent Quota	Determines the number of blocks of storage this UIC can own before VMS refuses to create new files or extend existing files. If the UIC has an Overdraft value greater than 0, a user with this UIC can retry the file operation (create or extend).	/PERMQUOTA
Overdraft	Determines the number of blocks above the permanent quota this UIC can own before VMS refuses to create new files or extend existing files. Therefore, the permanent quota plus the overdraft defines the total number of blocks available to a user on a volume.	/OVERDRAFT

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA SHOW [*,*] /DEVICE=DISK$USER

   UIC                 Usage          Permanent Quota      Overdraft Limit
[0,0]                  0              690                  200
[SYSTEM]               12047          13000                200
[VMS,BEYER]            11685          15000                200
[11,15]                56             56                   200
[VMS,CLARK]            16233          20000                200
[VMS,DORSEY]           13510          20000                200
[VMS,HARKINS]          18221          20000                200
[VMS,HUNT]             21060          30000                200
[11,340]               22905          30000                200
[VMS,DISALVO]          9021           18000                200
[VMS,TARGONSKI]        2425           4000                 200
[12,1]                 4              690                  200
[BEYER2]               142            144                  200
[GROUP21,ALBERT]       14137          20000                200
[21,10]                10             690                  200
[21,20]                2              690                  200
[GROUP21,EBERT]        5962           12000                200
[GROUP21,GALVIN]       3295           5000                 200
[GROUP21,TATAR]        32             2000                 200
[31,5]                 2              2                    200
[GROUP31,HARBO]        6117           10000                200
[GROUP31,CONNOR]       3261           8000                 200
[PAPISON]              666            690                  200
[CHERPAS]              19             690                  200
[GROUP101,ALCOCK]      29806          30000                200
[GROUP101,LUCAS]       27257          30000                200
[GROUP101,MASORS]      125            690                  200
[GROUP101,WILSON]      20968          25000                200
[123,321]              20             690                  200
[DATA_COMM,DELLA]      12931          20000                200
[DATA_COMM,LENTZ]      6341           20000                200
[200,3]                2              690                  200
[200,200]              60             690                  200
[DECNET]               78             690                  200
[J65,DOE]              4              100                  100

SYSMAN> EXIT
$
```

Example 2–8 Volume Quota File Records

Table 2–23 Displaying the Contents of a Volume Quota File

Operation	SYSMAN Command Format
Displaying the entry of a particular user	`DISKQUOTA SHOW [uic]`
Displaying the entries of all users with UICs in a particular group	`DISKQUOTA SHOW [group-number,*]`
Displaying the entries for all users	`DISKQUOTA SHOW [*,*]`
Displaying DISKQUOTA commands	`HELP DISKQUOTA`

NOTE
The disk volume usage recorded by the VMS operating system and displayed by SYSMAN includes some overhead. Therefore, the disk usage displayed by the SYSMAN command DISKQUOTA SHOW and the DCL command SHOW QUOTA is always greater than the disk usage displayed by the DCL command DIRECTORY/SIZE=ALLOCATED.

While the VMS operating system is running, it automatically updates the entries in the QUOTA.SYS files on volumes where you have enabled quotas. This update occurs to correct the usage counts whenever users acquire or release disk space on volumes where quotas are enabled.

A QUOTA.SYS file must exist on a volume for the VMS operating system to record the usage of that volume. You do not have to add a quota entry for a UIC to every quota file on your system. You may add an entry to one quota file and not to another. Also, the VMS operating system records usage on a per volume basis, so the usage counts for a UIC will typically be different on each volume.

A user with the EXQUOTA privilege has unlimited access to any volume where quotas are enabled, whether you have established limits for the user on that volume or not. When a user with this privilege creates or extends a file on a volume where quotas are enabled, the VMS operating system records that space usage in the quota entry for that user. If a quota entry does not exist on the volume for that user, the usage is not recorded. Therefore, if you allow a user to have the EXQUOTA privilege, you should add an entry for that user to each of your quota files to keep track of his or her use of space on those volumes.

To add an entry to a QUOTA.SYS file, run the SYSMAN utility, and enter the ADD command. Table 2–20 shows the syntax of SYSMAN's DISKQUOTA commands ADD, MODIFY, and REMOVE. Table 2–24 shows examples of the use of these commands.

After you add an entry for a UIC, user processes with that UIC may begin creating files on the volume. In Figure 2–3, the system manager adds a quota entry to the quota file on the DISK$DATA volume for the UIC [11,2]. After the entry has been added, the user can create files on that volume.

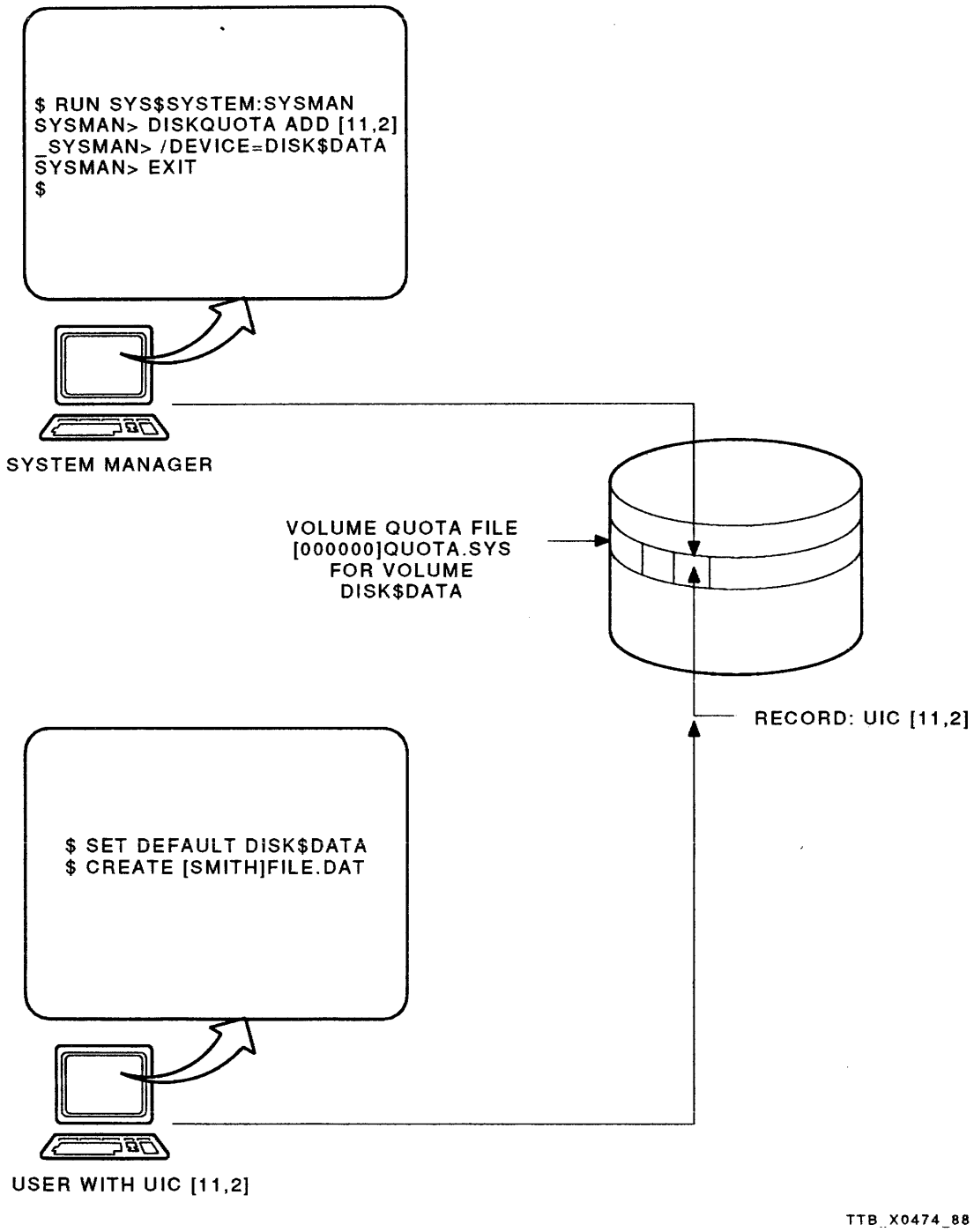


Figure 2–3 Adding a Quota Record to a Volume Quota File

The SYSMAN DISKQUOTA commands are executed immediately after you enter them, and SYSMAN immediately records your changes in the current QUOTA.SYS file (specified with the /DEVICE qualifier). Therefore, you do not have to exit SYSMAN to put your changes into effect.

Table 2–24 Managing Individual Records in the Volume Quota File

Operation†	SYSMAN Command Format
Adds a new entry, specifying values different from the default entry ([0,0])	`DISKQUOTA ADD uic [/PERMQUOTA=blksl] [/OVERDRAFT=blks2]`
Modifies an existing entry	`DISKQUOTA MODIFY uic [/PERMQUOTA=blksl][/OVERDRAFT=blks2]`
Modifies the entry for [0,0], used to supply default values for Permquota and Overdraft ([0,0], should never own any files)	`DISKQUOTA MODIFY [0,0] [/PERMQUOTA=blksl] [/OVERDRAFT=blks2]`
Modifies all entries for UICs in a particular group	`DISKQUOTA MODIFY [grp-num,*] [/PERMQUOTA=blksl] [/OVERDRAFT=blks2]`
Modifies all entries, including the default entry, [0,0]	`DISKQUOTA MODIFY [*,*] [/PERMQUOTA=blksl] [/OVERDRAFT=blks2]`
Removes an existing entry	`DISKQUOTA REMOVE uic`

†The SYSMAN utility performs all operations on the current QUOTA.SYS file. The current file is the one on your current default device if you did not specify one with the /DEVICE qualifier. Be sure to specify the proper volume for your command; otherwise, the most recently used /DEVICE qualifier sets the current file specification.

Establishing Quotas on an Existing Volume

Establishing quotas on a volume that has been in use requires more steps than for a newly created volume. These steps ensure that users of the volume are not adversely impacted during the quota creation procedure, and that they have suitable quota entries after the quota file has been created.

The quota file creation procedure for an existing volume requires the use of the SYSMAN command DISKQUOTA REBUILD. It is important to note that the REBUILD command **software write-locks** the volume during the rebuild process. This means that users are not able to create or extend files during the rebuild process.

Before creating the quota file, make sure that all users of the volume have been properly notified of the impending rebuild procedure. (Communicating with users is discussed in a later section of this module.) Table 2–25 contains detailed steps for properly creating a quota file on an existing volume called DISK$USER.

Table 2–25 Establishing Quotas on an Existing Volume

Steps	Commands	Comments
1	Notify users that DISK$USER will be unavailable	There are a number of ways to notify users of the volume. See the section **Communicating with User Processes** in this module.
2	`$ RUN SYS$SYSTEM:SYSMAN`	Invoke the SYSMAN utility. Make sure your current process has the OPER privilege to use the SYSMAN utility.
3	`SYSMAN> DISKQUOTA CREATE -` `_SYSMAN> /DEVICE=DISK$USER`	Creates the file DISK$USER:[000000]QUOTA.SYS and automatically enables quotas on the volume.
4	`SYSMAN> DISKQUOTA MODIFY [0,0] -` `_SYSMAN> /PERMQUOTA=10000 /OVERDRAFT=1000`	Sets the default entry values for the quota file on DISK$USER. Use appropriate values for `/PERMQUOTA` and `/OVERDRAFT` to reflect your management policy on the volume. Note that the `/DEVICE=DISK$USER` qualifier need not be specified, as the qualifier was properly specified in Step 3.

Table 2–25 (Cont.) Establishing Quotas on an Existing Volume

Steps	Commands	Comments
5	SYSMAN> DISKQUOTA REBUILD	Updates the newly created quota file to add existing UICs that own files on the DISK$USER volume. Note again that the /DEVICE=DISK$USER qualifier need not be specified in this case.
6	SYSMAN> EXIT	Exit from the SYSMAN utility.
7	Notify users that DISK$USER is available for use.	See the section **Communicating with User Processes** for possible techniques of user notification.

You can disable quotas on a volume at any time using the SYSMAN command DISKQUOTA DISABLE. Temporarily disabling quotas on a volume can be useful for certain situations, such as when you need to run a test program that creates large scratch files belonging to several UICs. You can later enable them again with the SYSMAN command DISKQUOTA ENABLE after the test is over and the scratch files have been deleted.

If you disable quotas for a period of time on a volume, the usage counts recorded in the volume's quota file will no longer be correct. Use the SYSMAN command DISKQUOTA REBUILD at any time to update the usage counts for all UICs on the volume, but be sure to follow the steps outlined in Table 2–25 to ensure that users are sufficiently safeguarded against the software write-lock that occurs during the rebuild procedure.

> NOTE
> Whenever a disk volume is unloaded improperly, for example, during a system failure, an automatic DISKQUOTA REBUILD will occur when it is remounted if a quota file exists on the volume (unless you include the /NOREBUILD qualifier when you remount it). The /NOREBUILD qualifier is discussed in Module 4.

For more information on DISKQUOTA commands, refer to the *VMS SYSMAN Utility Manual*. Additional information on disk space management can be found in the *Guide to Maintaining a VMS System*.

2.10 Laboratory Exercises

NOTE
You cannot perform these exercises unless you have write access to the quota file on your class volume. Check with your course administrator before you attempt the exercises.

1. Log in using your own account.
2. Run the SYSMAN utility (requires OPER privilege).
3. Enter the DISKQUOTA SHOW/DEVICE command, specifying your class volume name, and display all the quota records in the quota file for the class volume.
4. To observe the effects of disk quotas, perform the following:
   a. Display your current disk quota settings. Write them down.
   b. Delete your record from the quota file.
   c. Exit from the SYSMAN utility.
   d. Try to create a small text file by using either the CREATE command or a text editor.
   e. Reenter the SYSMAN utility and add a diskquota record for yourself. Specify the values you previously wrote down for permanent quota and overdraft. **Do not forget to specify your class volume name with the /DEVICE qualifier.**
   f. Exit from the SYSMAN utility.
   g. Enter the SHOW QUOTA command (from DCL level). Record the usage count. Create a small text file.
   h. Enter the SHOW QUOTA command again. Notice that your usage count has increased.
   i. Reenter the SYSMAN utility and display your diskquota record in the quota file for the class volume. Notice that your usage count has been increased here as well.
5. Modify your record to increase your permanent quota by 1000 blocks and your overdraft by 200 blocks.
6. Exit from the SYSMAN utility.
7. Set your default to SYS$SYSTEM.
8. Run the SYSMAN utility again. Enter the appropriate commands to display your record in the quota file for the class volume.

2.11 Solutions to Laboratory Exercises

1. No solution required.
2. `$ RUN SYS$SYSTEM:SYSMAN`
3. `SYSMAN> DISKQUOTA SHOW /DEVICE=CLASS_DISK [*,*]`

   Substitute the name of your class volume for CLASS_DISK.
4. Enter the following commands to observe the effects of disk quotas.
   a. `SYSMAN> DISKQUOTA SHOW [320,10]`

      Substitute your UIC for [320,10]. If the utility displays the alphanumeric form of your UIC, substitute it for [320,10]. For example, if the string form of your UIC is [GRP320,SMITH], you should enter the following command:

      ```
      SYSMAN> DISKQUOTA SHOW [GRP320,SMITH]
      ```

   b. `SYSMAN> DISKQUOTA DELETE [320,10]`
   c. `SYSMAN>EXIT`
   d. Note that you are unable to create files on the volume because you do not have a disk quota.
   e.
      ```
      $ RUN SYS$SYSTEM:SYSMAN
      SYSMAN> DISKQUOTA ADD /DEVICE=CLASS_DISK [320,10]
      ```

      Substitute the name of your class volume for CLASS_DISK and your UIC for [320,10].
   f. `SYSMAN>EXIT`
   g. `$ SHOW QUOTA`

      The usage count should be 0. The following is a short text file you might create.

      ```
      $ CREATE FILE.TXT
      This is a short text file.
      It should take up at least one block of space.
      CTRL/Z
      ```

h. `$ SHOW QUOTA`

i.
```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA SHOW [320,10]
```

5.
```
SYSMAN> DISKQUOTA MODIFY/DEVICE=CLASS_DISK [320,10] -
_SYSMAN> /PERMQUOTA=1500/OVERDRAFT=300
```

This assumes the original quota to be 500, with an allowed overdraft of 100.

6. `SYSMAN> EXIT`

7. `$ SET DEFAULT SYS$SYSTEM`

8.
```
$ RUN SYSMAN
SYSMAN> DISKQUOTA SHOW/DEVICE=CLASS_DISK [320,10]
```

2.11.1 Managing Disk Space Using DCL Commands

The system manager and the users can also enter DCL commands that help control the use of disk space. These DCL commands are SET FILE and SET DIRECTORY (see Tables 2–26 and 2–27).

With these commands, users or managers can limit the number of versions allowed per file or per directory. For example, to limit FILE.DAT to three versions, enter the following command:

```
$ SET FILE/VERSION_LIMIT=3 FILE.DAT
```

After you enter this command, only three versions of FILE.DAT can exist at any one time in your directory. When you create a new version of FILE.DAT, the VMS operating system deletes the oldest version automatically so only three versions are left. For example, if your directory contains three versions of FILE.DAT — FILE.DAT;1, FILE.DAT;2, and FILE.DAT;3 — the VMS operating system automatically deletes FILE.DAT;1 (the oldest version) when you create FILE.DAT;4.

To limit all the files in a new directory to three versions, use the command SET DIRECTORY:

```
$ SET DIRECTORY/VERSION=3 [SMITH]
```

SET DIRECTORY affects only the files created after it is entered. It does not affect currently existing files in a directory. To affect them, use the command SET FILE.

> NOTE
> You can override the effects of the command SET DIRECTORY for any given file in a directory by using the command SET FILE. For example, if the version limit for a directory is three versions, you can use SET FILE to increase the limit for FILE.DAT to five versions.

Another reason for using SET FILE and SET DIRECTORY is if you have to change the UIC of a user. You can change the UIC in the UAF and in the appropriate QUOTA.SYS file(s). Then you can modify the owner UIC of all the user's files and directories to match the user's new UIC.

For more information on modifying the characteristics of files, directories, and volumes, read the command descriptions for SET FILE, SET DIRECTORY, and SET VOLUME in the *VMS DCL Dictionary*.

Table 2–26 Controlling Files with DCL Commands

Operation	Command Format	Comments

Changing the ownership of a file.
 	
Remember that the VMS operating system records the use of disk space on a UIC basis, so changing the owner UIC of a file changes which disk quota entry the space is recorded under.
	`$ SET FILE/OWNER_UIC=uic -`
	`_$ file-spec`
	
	`$ SET FILE/OWNER_UIC=[320,63]`
	`- _$ INVENTORY.DAT`
		This operation requires SYSPRV privilege or a system UIC. One reason for using this would be if you forget the /OWNER qualifier when copying a file to a user's directory.

Establishing automatic version deletion for a particular file.	`$ SET FILE/VERSION_LIMIT=n -`
	`_$ file-spec`
	`$ SET FILE/VERSION_LIMIT=2 -`
	`_$ [MARTIN...]*.*`
		The VMS operating system automatically deletes the file with the lowest version number when you create a new version to keep the total number of versions equal to the version limit for the file.

Adding an ACL to a file or modifying an existing ACL on a file.	`$ EDIT/ACL/... filespec ; or`
	`$ SET ACL/OBJECT_TYPE=FILE -`
	`_$ /qualifiers -`
	`_$ file_spec`
	`$ SET ACL/OBJECT_TYPE=FILE -`
	`_$ /ACL=(IDENTIFIER=DEVELOPERS,`
	`-`
	`_$ ACCESS=WRITE) -`
	`_$ [ADMIN]PAYROLL.DAT`
		The keyword FILE specifies the file whose ACL is being edited.

Deleting unwanted versions of files.	`$ PURGE filespec /KEEP=n` `$ PURGE TEMP.DAT /KEEP=1`	The /KEEP qualifier is used to specify how many versions to retain. If no parameter is specified, the qualifier defaults to 1.

Table 2–27 Controlling Directories and Volumes with DCL Commands

Operation	Command Format	Comments

Changing the ownership of a directory.
 	
Changing the UIC changes which quota file entry the space is recorded under.
	`$ SET DIRECTORY/OWNER_UIC=uic -`
	`_$ directory-spec`
	`$ SET DIRECTORY/OWNER_UIC=[ENG] -`
	`_$ DISK$DESIGNS:[PRODUCT_X]`
		This operation requires SYSPRV privilege or a system UIC. One reason for using this would be if you forget the /OWNER qualifier when creating the directory for a user.

Establishing automatic version deletion for all files in a directory.	`$ SET DIRECTORY/VERSION_LIMIT=n -`
	`_$ directory-spec`
	`$ SET DIRECTORY/VERSION_LIMIT=3 -`
	`_$ DISK$MFG:[INVENTORY]`
		This sets a version limit on all files subsequently created in the directory. It does not affect previously created files. You can also use the SET FILE command to set a different limit for a particular file.

Adding an ACL to a directory file or modifying an existing ACL on a directory file.	`$ EDIT/ACL/... directory-spec ; or`
	`$ SET ACL/... -`
	`_$ directory-spec`
	`$ EDIT/ACL WORK1:[USERS]SMITH.DIR`
		Invokes the ACL editor, discussed later in this course in Module 8.

Disabling special protection on a directory file.	`$ SET FILE/NODIRECTORY file-spec`
	`$ SET FILE/NODIRECTORY -`
	`_$ [USERS]SMITH.DIR`
		This command enables you to delete a corrupted directory file. It is not intended for use on normal (valid) directory files.

Changing the ownership of a volume.
 	
Volumes have protection masks that are similar to file protection masks. A user who does not pass the volume protection mask cannot access any directories or files on the volume. If you change the owner UIC of a volume, you affect who will be able to access space on that volume.
	`$ SET VOLUME/OWNER_UIC=uic -`
	`_$ device-spec`
	`$ SET VOLUME/OWNER_UIC=[ADMIN] -`
	`_$ DISK$DATA`
		This operation requires the VOLPRO privilege.

2.12 Communicating with User Processes

Managers must be able to communicate with users and operators.

There are many ways of communicating on the system. One way is to set up a common directory or subdirectory that provides a common location to pass suggestions, ideas, new programs, command procedures, common problems, complaints, etc. A common directory relieves some of the burden of distributing information.

The VMS operating system provides other methods through the use of a number of programs and DCL commands. Table 2–28 lists the communication methods available on the VMS system.

Table 2–28 Communication Methods

Communication	Method
Operator/user direct dialogue	Phone utility
System-wide distribution of messages and information	Mail utility
User requests to the operator	REQUEST command MOUNT command
Operator responses to user requests, or short messages to users	REPLY command

2.12.1 Handling Requests for Operator Assistance

The **operator communication process**, OPCOM, coordinates all requests for operator assistance and all operator responses on a VMS system. OPCOM also maintains the operator's log, a file containing a record of these and other system-related messages.

Requests for operator assistance can be a result of the following:

- Explicit user commands (such as MOUNT or REQUEST/REPLY)
- I/O events (such as reading or writing up to the end of a tape reel)

Requests will be sent to an **operator terminal** depending on the category of the request. Any terminal can be designated as an operator terminal by using the REPLY/ENABLE command:

```
$ REPLY/ENABLE[=(keyword[,...])] terminal-spec
```

For example, the following command enables TXG6: to receive all operator-related messages for printers and tapes:

```
$ REPLY/ENABLE=(PRINTER,TAPES) TXG6
```

An operator terminal can also be disabled from having messages from one or more categories sent to it by using the REPLY/DISABLE command. For example, to disable printer-related messages from the terminal shown in the above example, use the following command:

```
$ REPLY/DISABLE=PRINTER TXG6
```

> NOTE
> OPER privilege is required to execute the REPLY/ENABLE and REPLY/DISABLE commands.

If the terminal you are using is enabled for one or more categories, you can obtain a list of which categories are enabled by using the REPLY/STATUS command. Table 2–29 lists the operator categories available for use with either the REPLY/ENABLE or REPLY/DISABLE commands.

Table 2–29 Operator Categories Enabled/Disabled with the REPLY Command

Category Keyword	Function
CARDS	Displays messages sent to the card readers.
CENTRAL	Displays messages sent to the central system operator.
CLUSTER	Displays messages from the connection manager pertaining to cluster state changes.
DEVICES	Displays messages pertaining to mounting disks.
DISKS	Displays messages pertaining to mounting and dismounting disk volumes.

Table 2–29 (Cont.) Operator Categories Enabled/Disabled with the REPLY Command

Category Keyword	Function
NETWORK	Displays messages pertaining to networks; the keyword CENTRAL must also be specified to inhibit network messages when disabling network messages with qualifier /DISABLE=NETWORK.
OPER1 through OPER12	Displays messages sent to operators identified as OPER1 through OPER12.
PRINTER	Displays messages pertaining to print requests.
SECURITY	Allows messages pertaining to security events. Requires SECURITY privilege.
TAPES	Allows messages pertaining to mounting and dismounting tape volumes.

Table 2–30 lists events that result in requests to OPCOM, which operators are notified by default, and what action they normally take.

If users perform operator functions themselves, they do not typically use the commands REQUEST or REPLY. However, they should learn how to respond to requests from the MOUNT command. The required response may be typing a command at a terminal, mounting a disk or tape volume, or both. (The MOUNT command is discussed in more detail in Module 4.)

Table 2–30 Events Requiring Operator Assistance

Event Causing Request	Operator Category Notified	Operator Action	Comments
MOUNT commands	DISKS TAPES	Mount a disk or tape volume.	By default, all MOUNT commands request operator assistance unless the appropriate volume is already loaded on the drive.
REQUEST/REPLY commands	All operators or only the group(s) of operators you specify. Group names include DISKS, TAPES and OPER1.	Perform the operation (or not), then respond to the user.	The user is waiting for a reply from the operator, and can do no other work until a reply is received.
I/O event — any command that reads a file on a tape	TAPES	Mount a new tape reel for the file system when it is reading a multireel volume set. Mount a new tape reel for the file system when it is writing a multireel volume set.	The operator receives a message from the Mount utility to mount another volume. The message looks similar to the message received from the MOUNT command when a user is attempting to mount a tape that is not loaded.

While one of the requests in Table 2–30 is pending, the user cannot issue another command. The user's terminal will not display the DCL prompt until the request is either satisfied or aborted. Either the user or the operator may abort the request. Example 2–9 shows a dialogue between a user and an operator in which the user issues the REQUEST/REPLY command, waits for an operator to respond, and finally cancels the request.

```
$
$ REQUEST/REPLY "Please respond to this message"
%OPCOM-S-OPRNOTIF, operator has been notified, waiting...  20:18:40.03
❶ %OPCOM-S-OPREPLY,
Hold on just a minute, please.
20:20:46.88, request 13578 is pending by operator _SUPER$TTD6:

%OPCOM-S-OPRNOTIF, operator has been notified, waiting...  20:20:46.99
❷ *CANCEL*
%OPCOM-I-RQST_PROMPT, REQUEST - Enter message or cancel request with control/Z.
❸ REQUEST - Message? *EXIT*
%OPCOM-S-OPREPLY,
%%%%%%%%%%  OPCOM  24-APR-1989 20:21:03.89  %%%%%%%%%%
Request 13578 was canceled
❹ $
```

Example 2–9 Using the REQUEST/REPLY Command

Notes on Example 2–9:

❶ Operator entered the command:

```
$ REPLY/PENDING=13578 "Hold on just a minute, please."
```

❷ User pressed CTRL/C.

❸ User pressed CTRL/Z instead of a message.

❹ User receives the DCL prompt and can continue working.

Example 2–10 shows a different dialogue where the operator aborts the request by entering the command:

```
$ REPLY/ABORT=14602 "There is no such tape"
```

Enter the REQUEST/REPLY command only when you require an operator to respond. To send a message to an operator that does not need a response, simply use the REQUEST command.

```
$ REQUEST/REPLY "Please mount the tape labeled MYBACKUP"
%OPCOM-S-OPENOTIF, operator has been notified, waiting...   20:25:58.64
%OPCOM-S-OPREPLY,
There is no such tape
20:26:20.81, request 14602 was aborted by operator _SUPER$TTD6:
$
$
```

Example 2–10 Operator Aborts a Request

```
                          ❶
%%%%%%%%%%  OPCOM  24-APR-1989  19:00:15.00  %%%%%%%%%%
❷Request 5377, from user JONES on CACAO     ❸
❹_CACAO$TWA18:, Please connect remote sensor 3A
$                         ❺
```

Example 2–11 Operator Receives Message from REQUEST/REPLY Command

```
                          ❶
%%%%%%%%%%  OPCOM  24-APR-1989  19:00:57.60  %%%%%%%%%%
❷Request 6401, from user SMITH on FUDGE     ❸
_FUDGE$TWA16:, Please mount volume DATA15 in device_MUA0:
$                         ❺
```

Example 2–12 Operator Receives Message from MOUNT Program

```
                          ❶
%%%%%%%%%%  OPCOM  24-APR-1989  17:01:15.00  %%%%%%%%%%
Message from user GREEN on TRUFFL     ❸
❹ _TRUFFL$TTB2:, Please check the line printer
$                         ❺
```

Example 2–13 Operator Receives Message from REQUEST Command

```
                          ❶
%%%%%%%%%%  OPCOM 24-APR-1989  19:02:16.92  %%%%%%%%%%
Message from user AUDIT$SERVER on SUPER  ❸
Security alarm (SECURITY) and security audit (SECURITY) on SUPER,
system id: 64063 / System UAF record modification
Event time:             24-APR-1989 19:02:16.82
PID:                    21E0009A
Username:               DUFFY
Image name:             SUPER$DUA0:[SYS10.SYSCOMMON.][SYSEXE]AUTHORIZE.EXE
Object name:            SYS$COMMON:[SYSEXE]SYSUAF.DAT;1
Object type:            file
User record added:      SMITH
Fields modified:        FLAGS,PWDLIFETIME,UIC
                          ❺
                        ❶
%%%%%%%%%%  OPCOM 24-APR-1989  19:02:37.41  %%%%%%%%%%
Message from user AUDIT$SERVER on SUPER  ❸
Security alarm (SECURITY) and security audit (SECURITY) on SUPER,
system id: 64063 / System UAF record modification
Event time:             24-APR-1989 19:02:37.32
PID:                    21E0009A
Username:               DUFFY
Image name:             SUPER$DUA0:[SYS10.SYSCOMMON.][SYSEXE]AUTHORIZE.EXE
Object name:            SYS$COMMON:[SYSEXE]SYSUAF.DAT;1
Object type:            file
User record modified:   SMITH
Fields modified:        PASSWORD
                          ❺
$
```

Example 2–14 Operator Receives Two Messages from Authorize Utility

Notes on Examples 2–11, 2–12, 2–13, and 2–14:

❶ Date and time of request or message

❷ Request identification number, used by the operator to identify the request when responding (only present if reply is required as in Examples 2-11 and 2-12).

❸ User name of process making request

❹ Terminal name if the process making the request is interactive (only present if request came from REQUEST or REQUEST/REPLY as in Examples 2–11 and 2–13).

❺ Message text

To respond to user requests, you must log in and enable your terminal as an operator terminal. Enabling your terminal and responding to operator requests requires OPER privilege.

NOTE
The system console terminal (OPA0) is automatically enabled as an operator terminal when the system is initialized, and receives all messages sent to operators whether it is logged in or not.

Table 2–31 explains how to respond to a request for operator assistance. Note that each response specifies a request identification number. You obtain this number from the request message you are responding to. Example 2–15 shows user and operator terminal activity during a typical communication session.

Table 2–31 Providing Operator Assistance

Circumstance	Command Format	Comments
User-requested function	`$ REPLY/TO=request-id -` `_$ "message-text"`	Users request various functions. Use the REPLY command with an appropriate message to respond.
User enters MOUNT command without loading a disk or tape volume in drive	No command required. Just load the volume requested on the drive.	The Mount utility sends a message to the operator terminal(s). The message includes the name of the volume requested.
User enters MOUNT command without the volume first being loaded. You attempt to load the volume on the drive, but fail. You decide to load the volume on an alternate drive (MTA2).	`$ REPLY/TO=request-id -` `_$ "Reissue mount using MTA2"`	Tell the user where you are loading the volume. The user must enter the MOUNT command again.
User needs to read information that spans more than one volume. You must load the next volume when the system requests it.	`$ REPLY/TO=request-id -` `_$ "volume-label"`	Respond to the system message after you have loaded the volume. Specify the volume's label in the message.

User needs to write more information than one volume can contain. You must load another volume when the system requests it.	`$ REPLY/INITIALIZE_TAPE= -`
	`_$ request-id -`
	`_$ "tape-label"`
		Use the /INITIALIZE_TAPE qualifier if the tape has been initialized before. The VMS operating system checks the tape's protection code. If you are not allowed access according to the code, you need the VOLPRO privilege to gain access.

	`$ REPLY -`
	`_$ /BLANK_TAPE=request-id -`
	`_$ "tape-label"`
		Use the /BLANK_TAPE qualifier for new tapes. You need VOLPRO privilege because it bypasses volume protection checking.

Table 2–31 (Cont.) Providing Operator Assistance

Circumstance	Command Format	Comments
Cannot perform user-requested function; must abort request	`$ REPLY/ABORT=request-id -` `_$ "message-text"`	Typically used when you cannot locate or use a resource requested by the user, (such as a volume, a drive, paper, or ribbon). Abort the request to cancel the repeated messages sent to the operator terminal. Find the resource or use the REPLY command or the Mail utility to explain the situation more completely.
May be able to perform requested function; want to notify interactive user, but leave request pending	`$ REPLY/PENDING=request-id -` `_$ "message-text"`	Since the request is pending, the user does not regain control of his or her own terminal. The user regains control (returns to the DCL prompt) by canceling the request or by waiting until the operator completes it.

```
Tapes:  $ REPLY/ENABLE=(TAPES,OPER1)
        %%%%%%%%%%%  OPCOM  24-APR-1989  19:28:18.57  %%%%%%%%%%%
        Operator _SUPER$TTA1: has been enabled, username TAPEOPR

        $
        %%%%%%%%%%%  OPCOM  24-APR-1989  19:28:18.73  %%%%%%%%%%%
        Operator status for operator _SUPER$TTA1:
        TAPES, OPER1

        $

Jones:  $ REQUEST/TO=TAPES/REPLY "Please mount tape 362F"
        %OPCOM-S-OPRNOTIF, operator has been notified, waiting... 19:31:18.65

Tapes:  $
        %%%%%%%%%%%  OPCOM  24-APR-1989  19:31:18.67  %%%%%%%%%%%
        Request 9482, from user JONES on SUPER
        _SUPER$TTA2:, Please mount tape 362F

        $
        %%%%%%%%%%%  OPCOM  24-APR-1989  19:31:18.67  %%%%%%%%%%%
        Request 9482, from user JONES on SUPER
        _SUPER$TTA2:, Please mount tape 362F

        $ REPLY/PENDING=9482 "You are second in line"

Jones:  %OPCOM-S-OPREPLY,
        You are second in line
        19:31:48.82, request 9482 is pending by operator_SUPER$TTA1:

        %OPCOM-S-OPRNOTIF, operator has been notified, waiting... 19:31:49.05

Tapes:  $ REPLY/TO=9482 "Tape 362F is now mounted"
        Tape 362F is now mounted
        19:39:18.48, request 9482 was completed by operator _SUPER$TTA1:

        $

Jones:  %OPCOM-S-OPREPLY,
        Tape 362F is now mounted
        19:39:18.48, request 9482 was completed by operator _SUPER$TTA1:

        $ REQUEST/TO=TAPES "Thank you"

Tapes:  $
        %%%%%%%%%%%  OPCOM  24-APR-1989  19:39:43.16  %%%%%%%%%%%
        Message from user JONES on SUPER
        _SUPER$TTA2:. Thank you

        $ REPLY/DISABLE=(TAPES)
        %%%%%%%%%%%  OPCOM  24-APR-1989 19:40:02.16  %%%%%%%%%%%
        Operator status for operator _SUPER$TTA1:
        OPER1

        $
```

Example 2–15 Request-Reply Interaction Between User JONES and User TAPES

In addition to sending messages to users when you reply to their requests, you can broadcast messages to all terminals or to particular terminals. Use the REPLY command for this function as well. Table 2–32 shows how to send such messages.

Table 2–32 Sending Messages to Users

Operation†	Command Format and Examples
Sending a message to all terminals on the system	`$ REPLY/ALL "message-text"` `$ REPLY/ALL/BELL "DISK$USER will be unavailable at noon."`
Sending a message to all terminals where users are logged in	`$ REPLY/USERNAME "message-text"` `$ REPLY/USERNAME "Policy Meeting in 10 minutes"`
Sending a message to specified users	`$ REPLY/USERNAME=(name [,...]) "message-text"` `$ REPLY/USERNAME=(JKMARTIN,BECKER) "DISK$DATA is available"`

Sending a message to a specified terminal or terminals	`$ REPLY/TERMINAL=(term [,...]) "message-text"`
	`$ REPLY/TERMINAL=(TXQ2,TXQ3) -`
	`_$ "Terminal repairperson due at 3pm!"`

Sending a message about system shutdown‡	`$ REPLY/SHUTDOWN "message-text"` `$ REPLY/SHUTDOWN "System will be down for PM at 2."`
Sending an urgent message‡	`$ REPLY/URGENT "message-text"` `$ REPLY/URGENT "Everyone off the system NOW!!"`

†To ring or buzz the terminal receiving the message, use the additional /BELL qualifier. More bells ring for URGENT and SHUTDOWN messages than for other types of messages.
‡ Use the /ALL qualifier. URGENT and SHUTDOWN messages are NOT sent to user terminals that have the command SET TERMINAL/NOBROADCAST.

2.12.2 The REPLY Command

The REPLY command has a large number of qualifiers that can be divided into the following categories according to their function:

- Responding to requests
- Sending messages and information
- Controlling operator terminals and the log file

Table 2–33 lists some qualifiers to the REPLY command in these categories.

Table 2–33 Qualifiers to the REPLY Command

Function	Qualifier
Responding to Requests	
Lists the requests that have not been sent a final message	`/STATUS`
Sends a final response to a request	`/TO=request-id`
Sends a response to a request, but not the final response	`/PENDING=request-id`
Cancels the request	`/ABORT=request-id`
Sends the label of an initialized tape as the final response	`/INITIALIZE_TAPE=request-id`
Sends the label of a blank tape as the final response	`/BLANK_TAPE=request-id`

Table 2–33 (Cont.) Qualifiers to the REPLY Command

Function	Qualifier
Sending Messages and Information	
Sends a message to a specific terminal	/TERMINAL=(term [,...])
Sends a message to all users presently logged in to the system, or to specific users	/USERNAME /USERNAME=(name [,...])
Sends a message to all terminals whether a user is logged in or not	/ALL
Sends an urgent message to all users	/URGENT
Sends a shutdown message to all users	/SHUTDOWN
Causes the terminal receiving the message to buzz or ring (add this to any of the above qualifiers)	/BELL
Requests that the originating terminal be notified when the message is successfully received	/NOTIFY
Controlling Operator Terminals and the Log File	
Sets up a nonconsole terminal as an operator's terminal	/ENABLE
Stops a terminal from being an operator's terminal	/DISABLE
Closes the current OPERATOR.LOG file and opens a new file with a higher version number	/LOG
Designates this terminal to be an operator's terminal only for the current interactive session.	/TEMPORARY
Lists the categories enabled for this terminal	/STATUS

2.12.3 The Operator's Log File

The operator's log file, SYS$MANAGER:OPERATOR.LOG, contains a textual record of requests, replies, and other system events of interest to operators and system managers.

Table 2–34 shows how to control and display the operator's log. You should print and delete old logs from time to time to conserve disk space. This is usually accomplished by placing the following command in the site-specific startup file:

```
$ PURGE/KEEP=2 SYS$MANAGER:OPERATOR.LOG
```

Customizing the site-specific startup file is discussed in Module 5.

Table 2–34 Controlling the Operator's Log

Operation	Command Format	Comments
Closing the operator's log and opening a new one	`$ REPLY/LOG`	Requires OPER privilege.
Closing the operator's log but not opening a new one	`$ REPLY/NOLOG`	Requires OPER privilege.
Printing the operator's log file	`$ SET DEFAULT SYS$MANAGER` `$ PRINT OPERATOR.LOG;n`	In this command format, "`;n`" indicates the version number.

2.12.3.1 Additional References

- For more information on controlling processes, read the command descriptions for SET PROCESS and STOP in the *VMS DCL Dictionary*.
- For more information on controlling files and volumes, read the command descriptions for SET FILE, SET DIRECTORY, and SET VOLUME in the *VMS DCL Dictionary*.
- For more information on responding to user requests for assistance, read the section on **Performing Disk and Magnetic Tape Operations** in the *Guide to Maintaining a VMS System*. Also read the command description for the REPLY command in the *VMS DCL Dictionary*.
- For more information on the operator's log, read the section on **Maintaining System Log Files** in the *Guide to Maintaining a VMS System*.

2.13 Laboratory Exercises

You need two terminals to perform the following exercises. Designate one terminal as an Operator terminal, and the other terminal as a User terminal. The process running on the Operator terminal must have OPER privilege.

1. Enable the Operator terminal so that it will receive messages sent to **one** of the operator classes OPER1 through OPER12. (Ask your course administrator which operator class you should use.) Use the /TEMPORARY qualifier.
2. From the Operator terminal, use the REPLY command to send a message to all logged-in users, telling them that you are now on duty as an operator.
3. From the User terminal, use the REQUEST command to send a message that will be displayed on the Operator terminal. Observe the output on the Operator terminal.
4. Send another REQUEST message to OPER1 requiring a response. Observe the output on the Operator terminal.
5. Respond to the message from the Operator terminal, telling the User that you are working on the request but you do not have an answer yet.
6. Issue a REPLY command to display the status of outstanding requests made to your operator class.
7. After a while, issue an appropriate message from the Operator terminal and abort the User request.
8. Log out from the Operator terminal.
9. Send a message to OPER1 from the User terminal that does not require a response.
10. Does the message appear on the Operator terminal? Why or why not?
11. Log back in to the Operator terminal. Find out whether it is still enabled to receive operator messages.

2.14 Solutions to Laboratory Exercises

Substitute your assigned operator class for OPER1 in the following solutions.

1. `$ REPLY /ENABLE=OPER1 /TEMPORARY`
2. `$ REPLY /USERS /BELL "Janet S. is now on duty as OPER1."`
3. `$ REQUEST REPLY/TO=OPER1 "Please check LPA0."`

   You should see a message on the Operator terminal that includes this request.

4. `$ REQUEST /REPLY /TO=OPER1 "Is it OK to print from LPA0?"`

   You should see another message, but this one also contains a request identification number.

5. `$ REPLY /PENDING=123 "Still working on the problem."`

   (Substitute the appropriate request ID number for 123.)

6. `$ REPLY /STATUS`
7. `$ REPLY /ABORT=123 "LPA0 is disabled indefinitely. Use LPB0."`
8. No solution needed.
9. `$ REQUEST /TO=OPER1 "Please back up CLASS_DISK tonight."`
10. The message should not appear on the Operator terminal screen.
11. `$ REPLY /STATUS`

    You should see a message stating that an illegal operator request was made.

MANAGING QUEUES

3.1 Introduction

When you issue the PRINT commmand to print a file, all the system printers may already be in use. Because this condition occurs often, The VMS operating system maintains a list of all print requests in the order in which they occur. This ordered list is called a **print queue**, and the requests are called **print jobs**. The PRINT command places your job in the queue. When the system is ready to process your job, it passes it to the printer associated with the queue or the first available printer, which prints the file. Similarly, when you issue the SUBMIT command to execute a command procedure in a batch process, system resources may not be sufficient to support the immediate creation of a **batch process** for you. Therefore, the VMS operating system maintains a list of all batch requests in the order in which they occur. This ordered list is called a **batch queue**, and the requests are called **batch jobs**. When the system is ready to process your batch job, it creates a batch process and executes your command procedure in the context of that batch process.

System managers customize batch and print queues on their systems. They attempt to maximize the performance of the system by matching the system queue structure to the workload and resources of their system. Usually, the system manager includes queue customization commands in startup files, so queues are created automatically when he or she starts the system. Module 5 contains a more detailed discussion of startup files.

As the queues are used, various problems may require the system manager's intervention. For example, paper can get jammed in a line printer, printers can run out of paper, or a batch job may need to be aborted. In these and other cases, the system manager must enter queue management commands to control the jobs in a queue while fixing the problem.

3.2 Objectives

To share limited printing and CPU resources among users and processes, a system manager must be able to:

- Describe how the VMS operating system handles print and batch jobs
- Assess user requirements for batch and print facilities
- Perform the following queue management tasks:
  - — Select proper queue attributes and characteristics to match user requirements
  - — Create print and batch queues
  - — Restrict access to queues
  - — Modify the attributes and characteristics of queues
  - — Control queues
  - — Handle queue problems

3.3 Resources

The following publications are referenced in this module:

1. *VMS System Manager's Manual*
2. *Guide to Using VMS Command Procedures*
3. *VMS DCL Dictionary*
4. *Guide to VMS System Security*
5. *VMS Access Control List Editor Manual*
6. *Guide to Maintaining a VMS System*

3.4 Overview of Queue Facilities and Operations

The VMS operating system provides comprehensive facilities to dynamically manage print and batch queues and the jobs submitted to those queues. This section covers the following topics:

- The queue manager process (JOB_CONTROL)
- The system queue file (SYS$SYSTEM:JBCSYSQUE.DAT)
- Types of queues

3.4.1 The Queue Manager and the System Queue File

JOB_CONTROL is a system process that performs many system management and control tasks. Among these tasks is the management of the system-wide queue file.

The VMS operating system keeps all queue information in one file, SYS$SYSTEM:JBCSYSQUE.DAT. If you do not intend to use queues on your system, you can delete the queue file to save some space. Enter the following command to see how much space you will save:

```
$ DIR/SIZE=ALLOCATED SYS$SYSTEM:JBCSYSQUE.DAT
```

If you intend to use queues, you must create the queue file and start the queue manager with the following DCL command:

```
$ START/QUEUE/MANAGER
```

This command requires OPER and SYSNAM privileges. If the system queue file does not exist, this command creates it; otherwise, this command starts only the queue manager. Include this command in the SYSTARTUP_V5.COM file (or in another startup file) so queues will be started when the system is booted.

> NOTE
> Digital recommends the use of the qualifier /RESTART whenever you use the command START/QUEUE/MANAGER within system startup files, or when you need to restart the queue manager for any reason. (To prevent a looping condition, the job controller does not restart if it detects an error within two minutes of starting the queue manager.)

Modules 5 and 6 discuss startup files in more detail.

3.4.2 Types of Queues

The VMS operating system provides two general classes of queues: **execution** and **generic** queues. An execution queue accepts either batch or print jobs for processing, depending on how the queue is initialized. A generic queue holds jobs until they are transferred to an assigned execution queue for processing. These two queue classes are further categorized into various **types** of queues by the kind of job they accept and the type of device to which the output is directed. Print execution queues are sometimes called **physical** queues.

Users submit jobs to any one of three types of queues - execution, generic, or logical. The action the system takes on their jobs depends on the type of queue and its restrictions (see Table 3–1).

Table 3–1 Types of Queues

Queue Type	Action
Execution	Executes the job.
Generic	Moves jobs to specified execution queues when resources for executing the job are available.
Logical	Moves jobs to execution queues when the manager sets up an association between a logical queue and an execution queue. The association does not have to be permanent and it does not always have to be made with the same execution queue.

3.4.2.1 Execution Queues

An execution queue performs the actual processing of the job. There are two types of execution queues:

- Batch execution queues
- Output execution queues

A **batch execution queue** can accept (process) only batch jobs. A batch job executes as a detached process that sequentially runs one or more command procedures. More detail on how the VMS operating system handles batch jobs is discussed later in this module.

An **output execution queue** typically accepts print jobs for processing by an independent process called a **symbiont**. There are three types of output execution queues:

- Printer execution queues
- Terminal execution queues
- Server execution queues

Of these three types, printer and terminal execution queues are most commonly found on VMS systems. Printer and terminal execution queues use symbionts that direct output to line printers and terminal printers, respectively. The VMS operating system is distributed with a standard print symbiont for use with printer and terminal execution queues. The user can either modify this print symbiont, or develop an entirely new symbiont to manage the output to a particular printer.

A **server execution queue** uses a user-written or user-modified symbiont to process the files belonging to jobs in the queue. Although considered an "output execution queue," a server execution queue can be used by the user to queue files for processing of almost any type, not just for processing information for an output device.

This module addresses only the creation and management of printer and terminal execution queues. Additional information about server execution queues can be found in either the *VMS System Manager's Manual* or the *Guide to Maintaining a VMS System*.

3.4.2.2 Generic Queues

Generic queues are used to hold a job until an appropriate execution queue becomes available to process the job. The list of execution queues associated with a generic queue is defined when the generic queue is initialized. When an execution becomes available to process a job held in a generic queue, the job is requeued to the available execution queue. There are two types of generic queues, each corresponding to a particular type of output execution queue:

Generic batch queue	Directs jobs only to batch execution queues. These are typically used in VAXcluster systems to distribute the workload across several systems.
Generic output queue	Directs jobs to any of the three types of output execution queues: printer, terminal, or server.

3.4.2.3 Logical Queues

A **logical queue** is a specially defined output execution queue that has its output redirected. It is not a type of output execution queue in itself; rather, it holds a job until it can be transferred to a specific printer, terminal, or server queue. In a sense, a logical queue is something of a "holding" queue, where jobs held in the queue remain there until the logical queue is assigned to a specific output execution, and both queues are started.

> NOTE
> Logical queues can only be used with output execution queues (printer, terminal, or server) and **not** with batch queues. Logical queues are discussed in more detail later in this module.

3.5 How the VMS System Handles Print Jobs

You do not have to use print queues on your system. You can just allocate a printer and use the COPY command to send output to it. However, you cannot enter any other commands until the copy is done. You can also allocate a printer from within a program and use a write statement in the program to send output to it. However, your program cannot continue processing statements until the write is done.

This may be acceptable if you are the only one using the printer and you do not mind waiting. However, when several users are sharing the printer, the wait may be quite long because you cannot allocate the printer when another is using it.

Also, the printer does the print jobs on a first-come, first-served basis. If your job is more important than another user's job, but his or her job is already printing, you will have to wait (unless you turn off the printer and abort the job). Even when your job is short, if a larger job is sent first, you will have to wait.

The VMS operating system solves these waiting and scheduling problems by allowing you to create print queues. A queue is simply a list of jobs waiting to be printed. Users place jobs in print queues by entering the PRINT command. Once a job is in a queue, the VMS operating system decides when to print it. Depending on how the queue is set up, the VMS operating system may also decide where to print it.

Users can still send jobs to a printer with the COPY command or WRITE statements in programs, but only if the printer they are sending it to is **spooled**. The VMS operating system takes the output sent to a spooled printer, places it in a file, and submits the file as a job in the queue for the printer.

3.5.1 Print Job Scheduling

After jobs are placed in a print queue, the JOB_CONTROL process schedules the jobs according to their priority, submission time, and, in the case of print queues, the job size. It then delegates the actual execution of the job to one of its subprocesses, called **print symbionts** (see Figure 3–1).

You can see the JOB_CONTROL process and its print symbiont subprocesses in the output to the SHOW SYSTEM command (see Example 3–1 and the notes following it).

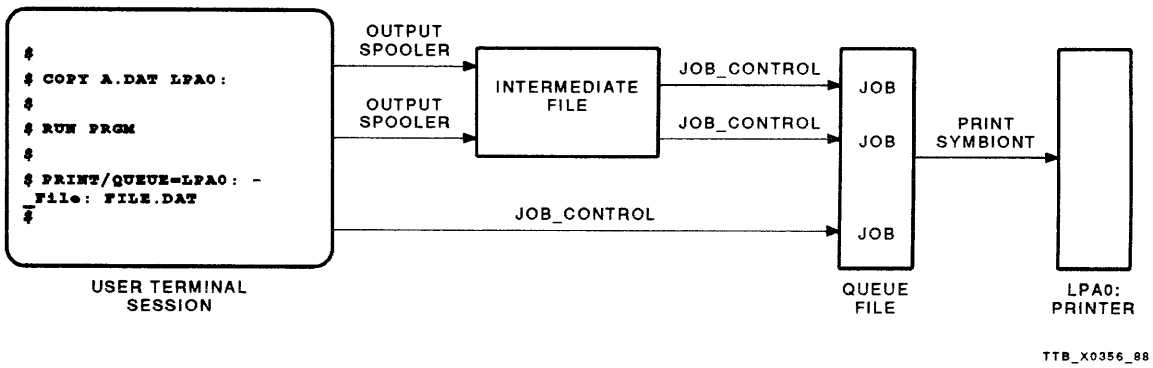


Figure 3–1 JOB_CONTROL Process Handles All Print Jobs

```
$ SHOW SYSTEM
VAX/VMS 5.2 on node BROWNY  19-APR-1989 16:55:43.02   Uptime  10 03:36:27
  Pid     Process Name    State  Pri      I/O       CPU        Page flts Ph.Mem
20200021 SWAPPER         HIB     16        0    0 00:00:21.96         0      0
20200263 DUFFY           CUR      4      328    0 00:00:10.49      1057    299
20200185 VPA_DC          HIB     15     4623    0 00:13:00.43      1374   1333
20200027 ERRFMT          HIB      8     9995    0 00:02:59.54        82    118
20200028 CACHE_SERVER    HIB     16      152    0 00:00:00.75        62     93
20200029 CLUSTER_SERVER  HIB      8       39    0 00:00:02.20       151    314
2020002A OPCOM           HIB      8     4321    0 00:02:13.76       645    211
2020002B AUDIT_SERVER    HIB     10       54    0 00:00:54.84      1300    223
❶ 2020002C JOB_CONTROL   HIB      8     3320    0 00:00:42.08       201    348
2020002D CONFIGURE       HIB     10      122    0 00:00:12.96       111    159
❷ 2020002E SYMBIONT_0001 HIB      6       85    0 00:00:03.92       670     46
2020002F SMISERVER       HIB      9      104    0 00:00:03.07       406    437
20200251 NETACP          HIB     10     1493    0 01:24:58.28   2321834   3500
20200112 EVL             HIB      5     1390    0 00:00:39.21     49642     38  N
202001B3 REMACP          HIB      9       59    0 00:00:00.56        80     50
20200075 WOODS           LEF      4    14551    0 00:09:09.52     37853    170
2020029A SPM_CAPACITY    LEF     24     1975    0 00:00:40.75       217    170
```

Example 3–1 JOB_CONTROL and Print Symbiont Processes

Notes on Example 3–1:

❶ The JOB_CONTROL process

❷ A print symbiont process

The JOB_CONTROL process executes the job with the highest queue priority first. If there are several jobs of the same size waiting at the same queue priority, they are executed in the order of submission. If the jobs are not the same size, the VMS system executes the smallest job within a queue priority group first (see Figure 3–2 and Example 3–2).

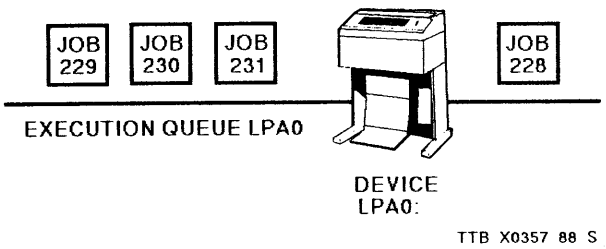


Figure 3–2 Print Queue

```
$
$ SHOW QUEUE LPA0/FULL
Printer queue LPA0
    /BASE_PRIORITY=4 /DEFULAT=(FLAG) /FORM=DEFAULT Lowercase
    /OWNER=[SYSTEM] /PROTECTION=(S:E,O:D,G:R,W:W)

  Jobname          Username     Entry  Blocks  Status
  -------          --------     -----  ------  ------
❶ ACTION           JONES          228       6  Printing
    Submitted 13-DEC-1987 12:02 /FORM=DEFAULT /PRIO=100
      _DRA1:[JONES]ACTION.COM;1 /COPIES=2

  NOTES            JONES          231      12  Pending
    Submitted 13-DEC-1987 12:15 /FORM=DEFAULT /PRIO=120
      _DRA1:[JONES]NOTES.TXT;1                    ❷
                                          ❸
  MEMO             JONES          230       1  Pending
    Submitted 13-DEC-1987 12:08 /FORM=DEFAULT /PRIO=100
      _DRA1:[JONES]MEMO.MEM;1

❹ MATH             JONES          229       6  Pending
    Submitted 13-DEC-1987 12:04 /FORM=DEFAULT /PRIO=100
      _DRA1:[JONES]MATH.LIS;1
$
$
```

Example 3–2 Scheduling Print Jobs

Notes on Example 3–2:

❶ Job 228 is currently executing. The JOB_CONTROL process examines the parameters of the pending jobs to determine which job it will print next.

❷ Since Job 231 has the highest priority of the pending jobs, it will be printed next.

NOTE
The priority of a job in a queue is limited by two system parameters:

- DEFQUEPRI - the default queue priority assigned to all print and batch jobs.
- MAXQUEPRI - the maximum queue priority any user can assign to a job (range is 0-255).

The default value of these parameters is 0. When the default is used, jobs receive the same queue priority as the base priority of the process (user) who submits them. For example, when the parameters are set to 0, jobs submitted by a user whose base priority is 4 are assigned a queue priority of 4 and jobs submitted by a user whose base priority is 6 are assigned a queue priority of 6. (Regardless of the values of these parameters, users with OPER or ALTPRI privilege can submit jobs at any priority using the /PRIORITY qualifier.)

❸ Job 230 is smaller than Job 229, and they have the same priority, so Job 230 will be printed third.

❹ Finally, Job 229 will be printed. However, if another job is submitted before Job 229 begins printing, the JOB_CONTROL process will examine the parameters of Job 229 and the new job to determine which job it will print first.

To learn more about how the VMS system handles print jobs, spooling, and print symbionts, consult the *Guide to Maintaining a VMS System* or the *VMS System Manager's Manual.*

3.6 Print Queue Operations

3.6.1 Types of Print Queues

There are four types of print queues. Two of these are terminal and printer execution queues. The other two are generic and logical queues. The system executes jobs in a printer execution queue on its associated printer; it executes jobs in a terminal execution queue on its associated terminal. The system does not execute jobs listed in generic or logical queues. Instead, it moves jobs from generic and logical queues to execution queues (automatically or when you enter certain queue management commands). Then, it executes them on the printer or terminal associated with the execution queue.

3.6.2 Creating Print Queues

Queues can be created either interactively or through the use of command files during system startup. Typically, a system manager will interactively create a new queue to verify its capabilities, then insert the proper queue creation and startup commands in the appropriate system startup command procedure.

You can create and start queues with one command, or you can create them with one command and start them later with another command (see Table 3–2).

Table 3–2 Initializing and Starting Queues

Command	Function
`$ INITIALIZE/QUEUE [/qualifiers] queue-name` `$ INITIALIZE/QUEUE /TERMINAL/ON=TXC2 LASER`	Creates the queue. If the queue is already running, this command has no effect. If a queue exists but is stopped, you can use this command to modify queue parameters. Jobs listed in the queue and new jobs will execute under the new parameters.
`$ START/QUEUE [/qualifiers] queue-name` `$ START/QUEUE /FORM=WIDE LPA0`	Starts a stopped queue. If the queue is already running, the system displays an error message.

`$ INITIALIZE/QUEUE/START [/qualifiers] -`
`_$ queue-name`
`$ INITIALIZE/QUEUE/START SYS$PRINT`
	Creates and starts a queue. Include this command for each queue in the SYSTARTUP_V5.COM file. If the queue is already running, this command has no effect.

To establish print queues, you must perform the operations, most of which require OPER privilege or ownership of the device (see Table 3–3).

Table 3–3 Establishing Print Queues

Step	Explanation
Set the physical attributes of each printing device.	Use the SET PRINTER command to set the attributes of printers. Use the SET TERMINAL command to set the attributes of terminals.
Spool each printing device.	Use the SET DEVICE/SPOOLED command. Note that, by default, the current default device (SYS$DISK) is used for the temporary files created by spooling. If your current default device (when you enter the SET DEVICE/SPOOLED command) is one in which you enforce disk quotas, spooling will fail for users without an entry in the disk's quota file.
Initialize and start an execution queue for each printing device (see Table 3–4).	If you do not want the system to automatically move jobs from generic queues to a certain execution queue, include the qualifier /NOENABLE_GENERIC when creating that execution queue (see Example 3–4). If you have more than one printing device, you may want to establish one for large jobs, one for medium jobs, and one for small jobs (see Table 3–5).
Initialize and start a generic print queue (normally, SYS$PRINT), if you have more than one printer with the same attributes on your system.	The system automatically moves jobs from the generic queue to print execution queues that were initialized without the /NOENABLE_GENERIC qualifier.
Initialize and start a generic terminal queue (giving it any name, such as TERMPRINT), if you have more than one terminal used as printers with the same attributes on your system.	This step is essentially the same as the previous step, except that you use terminal devices and specify the /TERMINAL qualifier when creating the queue, for example: `$ INITIALIZE/QUEUE/TERMINAL/GENERIC=(TTA1,TTC7) TERMPRINT`
If you want logical queues, initialize, assign and start them.	See Table 3-2 and Example 3-6 for several typical cases.

Table 3–4 Creating and Using Print Execution Queues

Operation	Creating a Printer Queue	Creating a Terminal Queue	Comments
Determine the device	`$ SHOW DEVICE L`	`$ SHOW DEVICE T`	List the devices and select one.

Set the device attributes	`$ SET PRINTER -`
	`_$ /UPPER LPA0`
		`$ SET TERMINAL -`
		`_$ /PERMANENT -`
		`_$ /NOTYPE_AHEAD -`
		`_$ /SPEED=2400 -`
		`_$ /NOBROADCAST -`
		`_$TTA3`
			Set attributes of printer or terminal to match its physical attributes or to force the printer to produce specific output. (For example, `/UPPER` causes all jobs to be printed in uppercase.) Terminals must have certain attributes set as shown (speed should be specified to match the terminal speed.)

Spool the device	`$ SET DEVICE -`
	`_$ /SPOOLED LPA0`
		`$ SET DEVICE -`
		`_$ /SPOOLED=WORK1 -`
		`_$ TTA3`
			Enables `COPY` commands and write statements for that device; you can specify an intermediate device or use the current default device (SYS$DISK).

Create and start the queue	`$ INITIALIZE/QUEUE -`
	`_$ /START/ON=LPA0 -`
	`_$ SYS$PRINT`
		`$ INITIALIZE/QUEUE -`
		`_$ /TERMINAL -`
		`_$ /START/ON=TTA3 -`
		`_$ SYS$PRINT`
			Assign it a different name than its device name by using the `/ON` qualifier. (By default, the queue name matches the printer name.)

List the device queues	`$ SHOW QUEUE/ALL -` `_$ /DEVICE`	`$ SHOW QUEUE/ALL -` `_$ /DEVICE`	Displays all execution queues.
Use the queue	`$ PRINT FILE.DAT`	`$ PRINT FILE.DAT`	Since the `PRINT` command sends files to the SYS$PRINT queue by default, and the name of the print execution queue for the LPA0 printer is SYS$PRINT, the first command prints FILE.DAT on LPA0. The second command specifies the TERM queue, so it prints the file on TTA3.

3.6.2.1 Creating Generic Print Queues

You can create more than one generic print queue. However, the system will move jobs from them to any available printer unless you specify that certain printers are associated with certain generic queues. To specify this, include the names of the printers when you initialize the generic queue. For example, to associate the printers LPA0 and LPB0 with the SYS$PRINT queue and the printers LPC0 and LPB1 with the LETTER queue, use the following commands:

```
$ INITIALIZE/QUEUE/GENERIC=(LPA0,LPB0) SYS$PRINT
$ INITIALIZE/QUEUE/GENERIC=(LPC0,LPB1) LETTER
```

If a generic queue is initialized using the qualifier /GENERIC, but no parameters are specified with the qualifier, the queue will feed any execution queue initialized with the qualifier /ENABLE_GENERIC.

When a print execution queue is created, it is automatically attributed as being able to receive jobs from generic queues that were created with the qualifier /GENERIC. You can override this attribute by using the qualifier /NOENABLE_GENERIC with the INITIALIZE/QUEUE command. See Table 3–5 for commands creating generic print queues.

Table 3–5 Creating and Using Generic Print Queues

Operation	Command	Comment

Create an execution queue for a printer.	`$ SET PRINTER/UPPER LPA0`
	`$ SET DEVICE/SPOOLED LPA0`
	`$ INITIALIZE/QUEUE/START -`
	`_$ LPA0`
		For example, the printer device LPA0.

Create an execution queue for another printer.	`$ SET PRINTER/UPPER LPB0`
	`$ SET DEVICE/SPOOLED LPB0`
	`$ INITIALIZE/QUEUE/START -`
	`_$ LPB0`
		For example, the printer device LPB0.

Create a generic print queue.	`$ INITIALIZE/QUEUE/START -` `_$ /GENERIC SYS$PRINT`	This queue receives default print jobs.
Use the generic print queue.	`$ PRINT FILE.DAT`	SYS$PRINT is the default queue for the `PRINT` command. In this example, SYS$PRINT is a generic print queue. The file is printed on LPA0 if it is available. If LPA0 is not available, and LPB0 is available, the file is printed on LPB0.

3.6.2.2 Creating Logical Print Queues

In many cases, you create a logical queue to hold jobs requiring certain types of paper or printer attributes. Users send jobs to these queues, and you can print them all at the same time during another shift after you have set the printer up properly.

You can use another method to accomplish the same purpose without creating logical queues by defining your own form types or printer attributes (see Table 3–6).

Table 3–6 Creating and Using Logical Queues

Operation	Command	Comment

Create an execution queue.	`$ SET PRINTER/UPPER LPC0`
	`$ SET DEVICE/SPOOLED LPC0`
	`$ INITIALIZE/QUEUE/START -`
	`_$ LPC0`
		Use LPC0 as the execution queue. Set the device to be spooled and to print only in UPPERCASE.

Create a logical queue.	`$ INITIALIZE/QUEUE AFTER5`	The `/START` qualifier is not valid for logical queues.
Use the logical queue.	`$ PRINT/QUEUE=AFTER5 -` `_$ FILE.DAT`	Jobs are listed in the queue but are not executed.
Associate the logical queue with the execution queue.†	`$ ASSIGN/QUEUE LPC0 AFTER5`	This tells JOB_CONTROL to "feed" LPC0 jobs from the AFTER5 queue once the AFTER5 queue is started.
Start the logical queue.	`$ START/QUEUE AFTER5`	The listed jobs are executed on the LPC0 printer.
Use the logical queue.	`$ PRINT/QUEUE=AFTER5 -` `_$ TEXT.DAT`	Job is listed in the AFTER5 queue and moved to the LPC0 queue to be printed when printer is available. Jobs are processed this way until the association is broken (with the `DEASSIGN/QUEUE` command).

†A logical queue does not have to be associated with an execution queue. It can be used simply as a holding queue for large jobs, jobs requiring special paper, or other types of jobs. The operator moves these jobs to specific execution queues to be printed when it is appropriate (see Table 3–15).

Figure 3–3 shows the relationship between several of the queues created in Tables 3–4, 3–5, and 3–6, and the physical printers available on a particular system. Example 3–3 shows the output to the SHOW QUEUE command that corresponds to Figure 3–3. Notice that the SHOW QUEUE command displays the queues in alphabetical order.

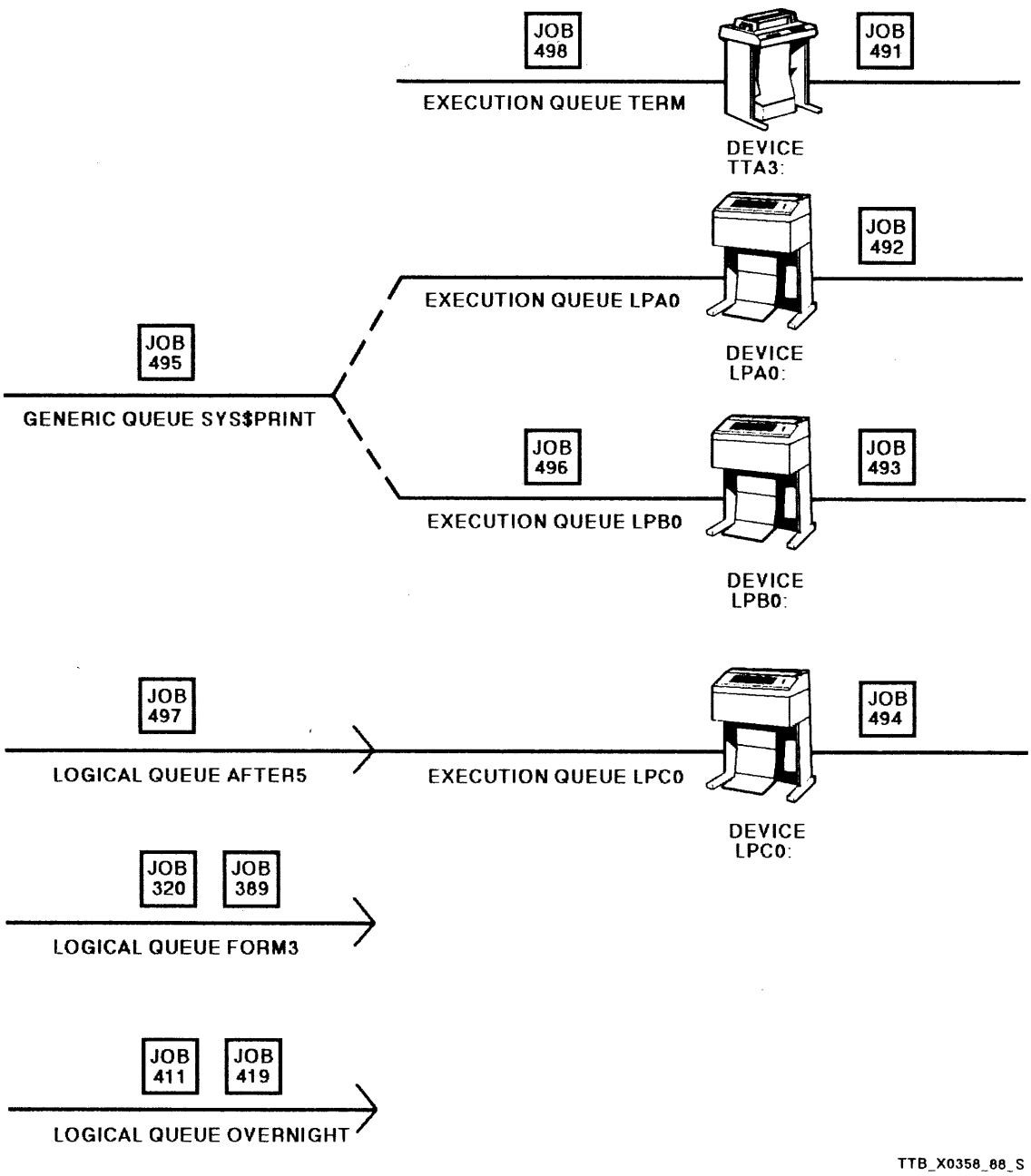


Figure 3–3 Current, Pending, and Holding Jobs

```
$
$ SHOW QUEUE/DEVICE/GENERIC
Logical queue AFTER5, assigned to LPC0

  Jobname         Username     Entry  Blocks  Status
  -------         --------     -----  ------  ------
  LATER           JONES          497       1  Pending

Printer queue FORM3, stopped

  Jobname         Username     Entry  Blocks  Status
  -------         --------     -----  ------  ------
  MATH            JONES          389       5  Pending
  TEST            JONES          320       7  Pending

Printer queue LPA0

  Jobname         Username     Entry  Blocks  Status
  -------         --------     -----  ------  ------
  MEMO            JONES          492       1  Printing

Printer queue LPB0

  Jobname         Username     Entry  Blocks  Status
  -------         --------     -----  ------  ------
  ACTION          JONES          493       1  Printing
  TABLES          JONES          496      21  Pending

Printer queue LPC0

  Jobname         Username     Entry  Blocks  Status
  -------         --------     -----  ------  ------
  FORTEST         JONES          494       1  Printing

Printer queue OVERNIGHT, stopped

  Jobname         Username     Entry  Blocks  Status
  -------         --------     -----  ------  ------
  LONG            JONES          419     400  Pending
  BIGJOB          JONES          411     478  Pending

Generic printer queue SYS$PRINT

  Jobname         Username     Entry  Blocks  Status
  -------         --------     -----  ------  ------
  MEMO            JONES          495       1  Pending

Terminal printer queue TERM, on TTA3:

  Jobname         Username     Entry  Blocks  Status
  -------         --------     -----  ------  ------
  PROG            JONES          491      10  Printing
  NOTES           JONES          498       9  Pending
$
$
```

Example 3–3 Queue Status Display of Current, Pending, and Holding Jobs

3.6.2.3 Automatic Queue Creation

To invoke the automatic creation and startup of queues at boot time, include the appropriate queue commands in the SYS$MANAGER:SYSTARTUP_V5.COM procedure. Example 3–4 shows a portion of a sample SYSTARTUP_V5.COM file containing commands to create print queues. The next section of this module contains a more detailed discussion of the commands shown in the procedure. By default, the system shutdown procedure SYS$SYSTEM:SHUTDOWN stops all queues.

More detailed discussions about startup and shutdown procedures are presented in Modules 5 and 6.

```
$ SET NOON
$ !
$ ! Start up the queue manager before issuing any other queue commands
$ !
$ START/QUEUE/MANAGER/RESTART
$ !
$ ! Define and start up printer queues
$ !
$ SET PRINTER/LOWER  LPA0
$ SET DEVICE/SPOOLED  LPA0
$ INITIALIZE/QUEUE/START/DEFAULT=(BURST,FLAG)  LPA0
$ !
$ SET PRINTER/LOWER  LPB0
$ SET DEVICE/SPOOLED  LPB0
$ INITIALIZE/QUEUE/START  LPA0
$ !
$ SET PRINTER/LOWER  LPC0
$ SET DEVICE/SPOOLED  LPC0
$ INITIALIZE/QUEUE/NOENABLE_GENERIC  LPC0
$ !
$ ! Define and start up a generic print queue
$ !
$ INITIALIZE/QUEUE/GENERIC/START  SYS$PRINT
$ !
$ ! Define a special logical queue and assign it
$ ! to execution queue LPC0, but don't start it yet
$ !
$ INITIALIZE/QUEUE  AFTER5
$ ASSIGN/QUEUE  LPC0  AFTER5
$ !
$ Define other logical queues
$ !
$ INITIALIZE/QUEUE  OVERNIGHT
$ INITIALIZE/QUEUE  FORM3
$
```

Example 3–4 Queue Startup Commands in SYSTARTUP_V5.COM

3.7 Laboratory Exercises

Use two terminals to do the following exercises. Use one terminal as an output device (printer). This can be a video or a hard-copy terminal. You will enter queue manipulation commands from the other terminal.

1. Set up one terminal as an output device.
   a. Log in to the terminal.
   b. Use the SHOW TERMINAL command to list the current characteristics of the terminal.
   c. Write down the device name of the terminal for future reference.
   d. Modify the characteristics of the terminal with the SET TERMINAL/PERMANENT command so that it can be used as an output device. (You will need LOG_IO and OPER privileges to do this.)
   e. Log out from the terminal.
2. Enter the following commands from the other terminal:
   a. Spool the output device.
   b. Create and start a queue for the output terminal. Give the queue a name using the /ON qualifier.
   c. Print a file at the output terminal using the queue you just created. Observe the output terminal to see if your job is displayed.
   d. Send a print job to your terminal execution queue, including the /HOLD qualifier in the command.
   e. Display the contents and characteristics of your terminal execution queue.
   f. Delete your job that is being held on the terminal execution queue.

3.8 Solutions to Laboratory Exercises

1. To set up a terminal as an output device, do the following:

   a. No solution needed.

   b.
   ```
   $ SHOW TERMINAL
   ```

   c. These solutions use the terminal name TTC5:

   d.
   ```
   $ SET TERMINAL /PERMANENT /NOTYPE_AHEAD -
   _$ /NOBROADCAST /SPEED=2400
   ```

   The /NOTYPE_AHEAD qualifier disables logins on this terminal. If you need to modify the terminal further after logging out, you can enter the SET TERMINAL/PERMANENT command at another terminal. For example, the following command modifies a characteristic of the TTC5 terminal:

   ```
   $ SET TERMINAL /PERMANENT /NOBROADCAST TTC5:
   ```

   e.
   ```
   $ LOGOUT
   ```

2. The following commands should eventually allow you to see your file displayed on the output terminal.

   a.
   ```
   $ SET DEVICE/SPOOLED TTC5:
   ```

   b.
   ```
   $ INITIALIZE/QUEUE/START/ON=TTC5: MYQUEUE
   ```

   Substitute the name of your output terminal for TTC5: and substitute your own queue name for MYQUEUE.

   c.
   ```
   $ PRINT/QUEUE=MYQUEUE FILE.TXT
   ```

   If you followed the instructions in this exercise carefully, you should see your file displayed on the output terminal. If your file does not appear, consult your course administrator.

   d.
   ```
   $ PRINT/QUEUE=MYQUEUE/HOLD FILE.TXT
   ```

   e.
   ```
   $ SHOW QUEUE/FULL/ALL MYQUEUE
   ```

   f.
   ```
   $ DELETE/ENTRY=369
   ```

   Substitute the entry number of your job for 369.

3.8.1 Monitoring Print Queues and Jobs

You can monitor the status of either entire queues or individual jobs. The amount of information displayed depends on your privileges and queue ownership rights.

3.8.1.1 Monitoring Queues

Use the DCL command SHOW QUEUE to monitor the status of entire queues, using the following format:

```
$ SHOW QUEUE [/qualifiers] [queue-name]
$ SHOW QUEUE /ALL/FULL LASER_Q
```

If you do not specify a qualifier or a queue name, the system displays the status of all queues on the system and all jobs owned by you (or all jobs if you have OPER privilege or E access to the queue). The SHOW QUEUE qualifiers allow you to select the type of queue information you want to display (see Table 3–7), or the amount of information you want to display (see Table 3–8).

Table 3–7 SHOW QUEUE Qualifiers for Displaying Types of Queues

Qualifier	Function
/BY_JOB_STATUS=status-type	Displays queues that contain jobs of a specified type of status. If no keyword is specified, by default the jobs of all status are displayed. The types are EXECUTING, HOLDING, PENDING, RETAINED, and TIMED_RELEASE.
/BATCH	Displays the status of batch execution queues.
/DEVICE=execution-type	Displays the status of output execution queues. If no keywords are specified, all types of output queues are displayed.
/GENERIC	Displays the status of generic queues.

Table 3–8 SHOW QUEUE Qualifiers for Displaying the Amount of Queue Information

Qualifier	Function
/ALL_JOBS or /ALL_ENTRIES	Displays information about all jobs for the selected queue.
/BRIEF	Displays a brief listing of information about job entries in the queue. The brief listing is the default when no qualifier is specified with the SHOW QUEUE command.
/FULL	Displays complete queue and job information, including any ACLs set for the queue.
/SUMMARY	Displays the total number of executing, pending, holding, retained, and time-released jobs.

You can combine certain qualifiers to further delineate the queue information you want displayed, for example:

```
$
$ SHOW QUEUE/SUMMARY/DEVICE=(PRINTER,TERMINAL)

Printer queue LPB0, mounted form DEFAULT
    5 holding,  2 timed release

Terminal queue LQ_PRINT, on TXA7:,
mounted form PORTRAIT_INDENTED (stock=DEFAULT)
    2 pending (8 blocks),   1 executing
$
$
```

See Table 3–9 for information on queue status codes returned by the SHOW QUEUE command.

You can further customize the type of queue information you want to monitor by writing a command procedure that uses the F$GETQUI lexical function. For more information about the F$GETQUI lexical function, see the *Guide to Using VMS Command Procedures*.

Table 3–9 Queue Status Codes Returned by the SHOW QUEUE Command

Status Code	Description
aligning	The queue manager is processing a START/QUEUE/ALIGN command.
device unavailable	Device to which the print symbiont is assigned is not available.
pausing	The queue manager is processing a STOP/QUEUE command.
paused	A STOP/QUEUE command has been executed.
resuming	The queue manager is processing a START/QUEUE command on a **paused** queue.
resetting	The queue manager is processing a STOP/QUEUE/RESET command.
operator service	A PRINT/OPERATOR command has been executed.
stalled	Print symbiont processing is temporarily halted due to a device-related problem.
stopping	The queue manager is processing a STOP command specified with either a /NEXT, REQUEUE, or RESET qualifier.
stop pending	Queue is **stopped** when current jobs have finished executing.
stopped	A STOP/QUEUE command specified with either a /NEXT, REQUEUE, or RESET qualifier has been executed.
starting	Queue has been started, but the print symbiont process is not yet active.

3.8.1.2 Monitoring Jobs

Use the DCL command SHOW ENTRY to monitor the status of individual jobs, using the following format:

```
$ SHOW ENTRY [/qualifiers] [queue-name]
```

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). You must have the OPER privilege or E access to the queue to display jobs owned by users outside your UIC group. The following command displays all jobs owned by user JONES:

```
$ SHOW ENTRY/USER=JONES

  Jobname         Username     Entry  Blocks  Status
  -------         --------     -----  ------  ------
  ACTION          JONES          228       6  Printing
  On terminal queue LASER_PRINT
```

The SHOW ENTRY qualifiers allow you to specify the type of job information you want displayed, similar to the SHOW QUEUE command qualifiers shown in Tables 3–7 and 3–8. Consult the *VMS DCL Dictionary* for a complete list of SHOW ENTRY qualifiers and their meanings.

The information displayed by the SHOW ENTRY command includes a status code for each job. The description of these codes in shown in Table 3–10.

Table 3–10 Job Status Codes

Status Code	Description
aborting	Executing job is terminating.
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.
waiting	Symbiont refuses the job.

3.8.2 Setting Print Queue Attributes

The system assigns certain default attributes to each queue when you create it, such as the:

- Owner (defaults to the user of the process creating the queue)
- Base priority
- Printer form definition
- Protection code

You can override these default attributes by defining different values for them when you create the queue. You can also define values for other attributes such as:

- Number of separation pages for print jobs
- Maximum and minimum allowed sizes of print jobs
- Printer characteristics

Many parameters exist to specify how a queue is to perform, or what jobs may be submitted to the queue. Most parameters can be specified as queue attributes during queue creation, while others must be specified after the queue has been created. There are times, particularly with print queues, when a system manager may need to modify an existing queue's attributes to reflect a change in execution requirements. See Table 3–11 for details on when to use the proper command to specify or modify a queue's attributes. Example 3–5 illustrates a situation where a queue is modified while running.

Table 3–11 Commands to Modify Queue Attributes at Certain Times

Command	When to Use
`INITIALIZE/QUEUE`	When the queue is being created (does not currently exist).

`SET/QUEUE`
`START/QUEUE`
`INITIALIZE/QUEUE`
	After the queue has been created, but is currently **stopped**.

`SET/QUEUE`	When the queue exists and is currently **running**. Note that not all parameters can be changed while the queue is running.

```
$
$ SHOW QUEUE/FULL LPA0
Printer queue LPA0
     /BASE_PRIORITY=4 /FORM=DEFAULT Lowercase /OWNER=[SYSTEM]
     /PROTECTION=(S:E,O:D,G:R,W:W)
$
$ SET QUEUE/SEPARATE=(BURST,TRAILER) LPA0
$
$ SHOW QUEUE/FULL LPA0
Printer queue LPA0
     /BASE_PRIORITY=4 /FORM=DEFAULT Lowercase /OWNER=[SYSTEM]
     /PROTECTION=(S:E,O:D,G:R,W:W) /SEPARATION=(BURST,TRAILER)
$
$ PRINT/HEADER MEMO.TXT
Job MEMO (queue SYS$PRINT, entry 349) started on SYS$PRINT
$
$
```

Example 3–5 Modifying a Running Queue

3.8.2.1 Specifying Separation Pages

Print job processing provides the ability to output sheets of paper, called **separation pages**, to make it easier to distinguish between individual jobs and files printed on a queue. There are two types of separation pages: **file separation pages** and **job separation pages**. File separation pages are output between the individual files that make up a print job, while job separation pages are output between complete print jobs. The types of file and job separation pages are illustrated in Figures 3-4 through 3-7.

Use the qualifier /SEPARATION with the appropriate options to either the INITIALIZE/QUEUE, SET QUEUE, or START/QUEUE commands to specify the types of job separation pages desired for a print queue (see Table 3–12).

The same options can be used with the qualifier /DEFAULT to specify default file separation pages, however you have slightly more control over the number of separation pages to be printed (see Table 3-13). By default, no job or file separation pages are assigned to the queue when it is created.

Table 3–12 Job Separation Page Options for the /SEPARATE Qualifier

Option	Function
[NO]BURST	Specifies a copy of the flag page is printed in such a way as to overprint the perforation between the preceding flag page. This makes it possible to determine job breaks in a stack of paper when viewed from the edge side of the paper. Note that if you specify a burst separation page, you do not need to specify a flag page, as it is automatically printed with the burst page.
[NO]FLAG	Specifies a page is printed preceding the job with the name of the user printed in large letters.
[NO]TRAILER	Specifies a single summary sheet is printed following a job, with the name of the user printed in large letters.

Table 3–13 File Separation Page Options for the /DEFAULT Qualifier

Option	Function
[NO]BURST[=keyword]	Specifies whether file burst pages will be printed. If the keyword is ALL (the default), a burst page is placed before each file in the print job. If the keyword is ONE, a burst page is placed before the first copy of the first file in the job. Note that if you specify a burst separation page, you do not need to specify a flag page, as it is automatically printed with the burst page.
[NO]FLAG[=keyword]	Specifies whether file flag pages will be printed. If the keyword is ALL (the default), a flag page is placed before each file in the print job. If the keyword is ONE, a flag page is placed before the first copy of the first file in the job.

Table 3–13 (Cont.) File Separation Page Options for the /DEFAULT Qualifier

Option	Function
`[NO]TRAILER[=keyword]`	Specifies whether file trailer pages will be printed. If the keyword is ALL (the default), a trailer page is placed at the end of each file in the print job. If the keyword is ONE, a trailer page is placed after the last copy of the last file in the job.

Users have control over which **file separation pages** are output, but only privileged users can control which **job separation pages** are output for a given queue. Job separation pages are sometimes referred to as "mandatory" queue attributes, referring to the fact that the user cannot alter the printing of separation pages surrounding a job on a given queue.

By setting file separation pages with the qualifier /DEFAULT, you can make print requests more convenient for the user community, since users will not have to specify these separation pages (unless they prefer a different combination of separation pages printed).

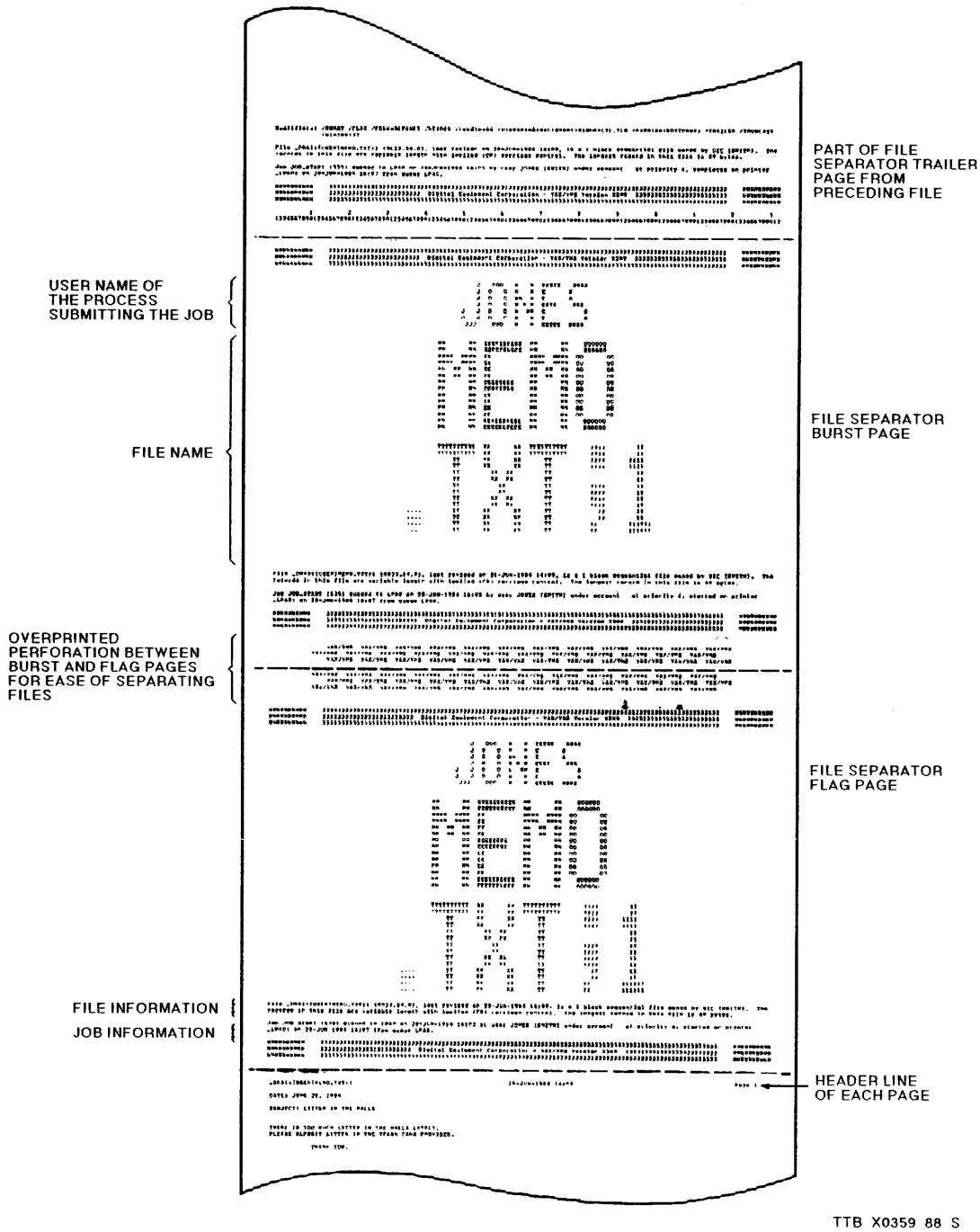


Figure 3–4 File Separation Burst and Flag Pages

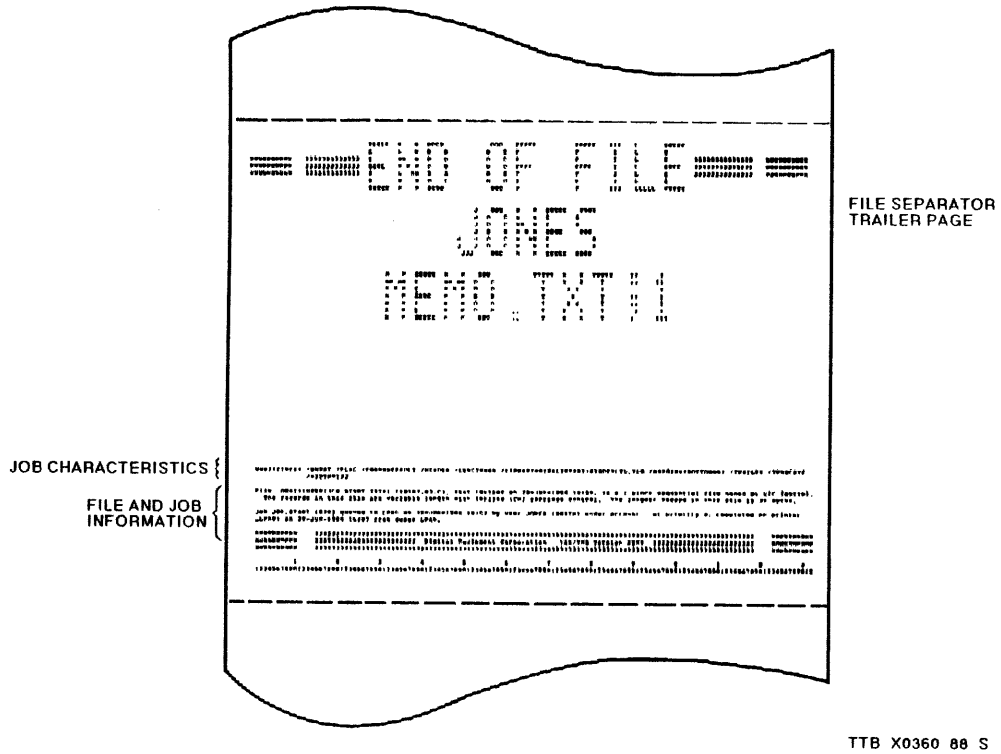


Figure 3–5 File Separation Trailer Pages

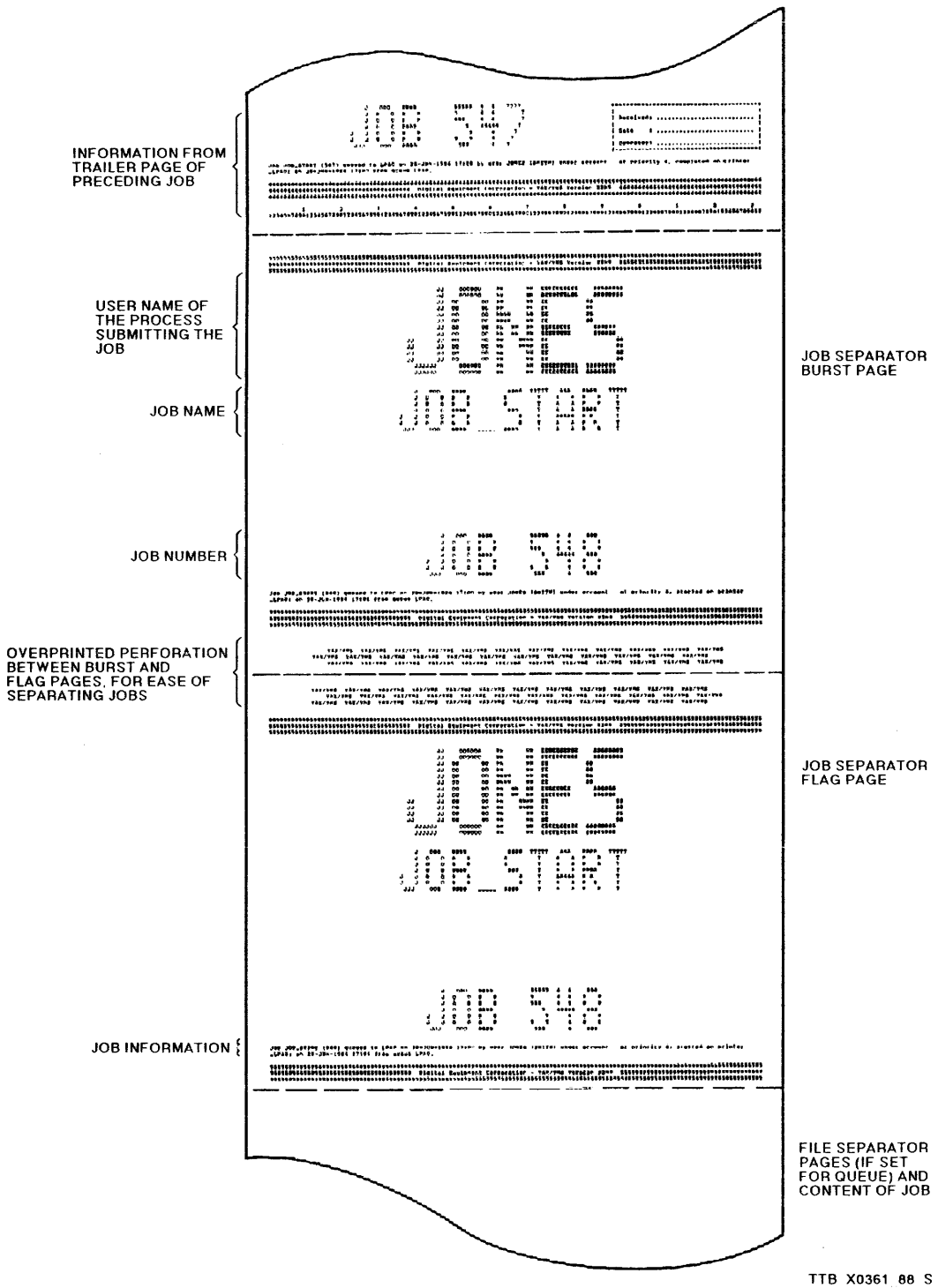


Figure 3–6 Job Separation Burst and Flag Pages

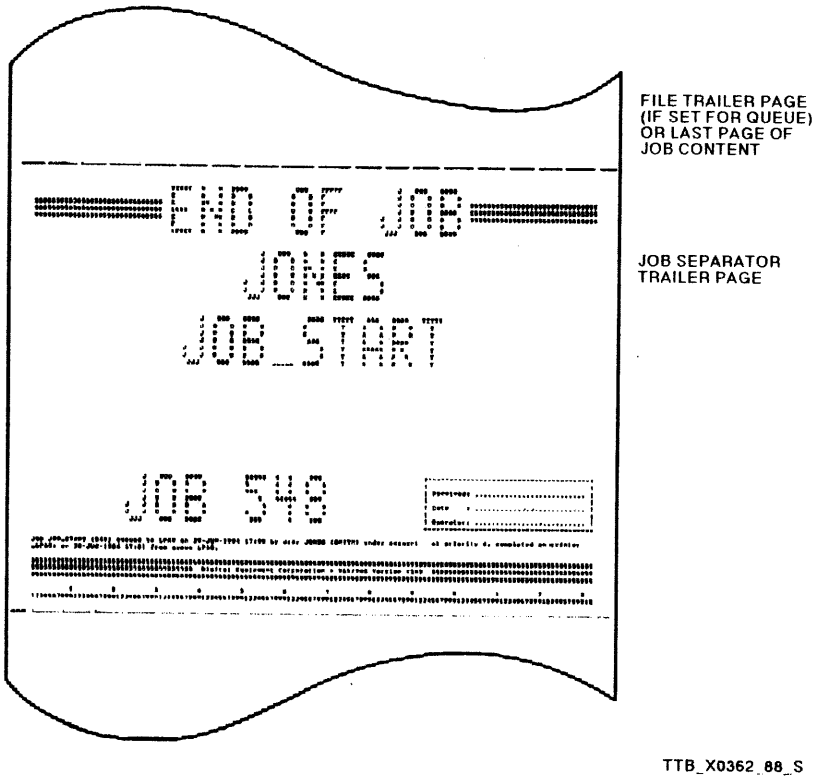


Figure 3–7 Job Separation Trailer Page

3.8.2.2 Limiting Print Job Sizes

If two or more jobs have the same priority, the job with the smallest size is executed first. To provide more control over which job sizes can print on which printers (or during which time of the day), you can set a queue to only process jobs that fit a job size criteria.

For example, if you have more than one printer of a particular type, you can set different limits to control which job sizes can execute on each printer. You may want one printer dedicated to "small" jobs (under 100 pages), another printer dedicated to "medium" jobs (100-500 pages), and yet another available for only "large" jobs (over 500 pages). Use the /BLOCK_LIMIT qualifer on either the INITIALIZE/QUEUE, START/QUEUE, or SETQUEUE commands to achieve this capability, as shown in Table 3–14.

Table 3–14 Setting Block Limits on Print Queues

Qualifier/Example	Comments
`/BLOCK_LIMIT=(minimum,maximum)`	Size of jobs submitted to queue must be within specified range.
`$ INITIALIZE/QUEUE/ON=LPA0 -` `_$ /BLOCK=(100,500) MEDIUM`	Jobs smaller than 100 blocks or larger than 500 blocks will not execute from the queue.
`$ INITIALIZE/QUEUE/ON=LPB0 -` `_$ /BLOCK=(500,"") LARGE`	Jobs smaller than 500 blocks will not execute from this queue. It is reserved for jobs larger than 500 blocks.
`$ INITIALIZE/QUEUE/ON=LPC0 -` `_$ /BLOCK=100 SMALL`	Jobs larger than 100 blocks will not execute from this queue. It is reserved for jobs smaller than or equal to 100 blocks. Notice that you do not need to include parentheses when you specify only an upper limit.

For more information on setting up print queues to match a system configuration, read the section on printer queues in either the *VMS System Manager's Manual* or *Guide to Maintaining a VMS System*.

For more information about modifying queue attributes and restricting the use of queues, read the information provided in the *VMS DCL Dictionary* for the INITIALIZE/QUEUE, SET QUEUE, and START/QUEUE commands.

3.9 Laboratory Exercises

1. Modify the default characteristics of the terminal execution queue you created in the last laboratory exercise so that all jobs sent to the queue are printed with flag and trailer pages.
2. Send a short text file to the output terminal and observe the display.
3. Initialize and start a batch queue on the system with a job limit of two. Name it MULTIBAT.
4. Change the job limit on MULTIBAT to four.
5. Display the characteristics of the queue after changing the job limit to check that the command was executed properly.

3.10 Solutions to Laboratory Exercises

1. `$ SET QUEUE/SEPARATE=(FLAG, TRAILER) MYQUEUE`

   Substitute the name of your queue for MYQUEUE.

2. `$ PRINT/QUEUE=MYQUEUE FILE.TXT`

   Substitute the name of your file for FILE.TXT.

3. `$ INITIALIZE/QUEUE/BATCH/START/JOB_LIMIT=2 MULTIBAT`

4. `$ SET QUEUE/JOB_LIMIT=4 MULTIBAT`

5. `$ SHOW QUEUE/FULL MULTIBAT`

3.10.1 Controlling Print Queues

The system manager should periodically monitor all jobs that execute on the system, especially print and batch jobs. Print jobs could attempt to execute on a malfunctioning printer, or on a printer with no paper. Printer ribbons can break and paper can jam. You must know what to do in each situation and others so jobs and information are not lost, and so system performance is not degraded.

You can modify the parameters of a queue by using the SET QUEUE or INITIALIZE/QUEUE commands. However, some situations call for other types of action, such as fixing the printer, adding paper, or replacing the printer ribbon. For these problems, you must have some control over the jobs in a queue.

You can stop or requeue a currently executing job without affecting the queue. In other situations, you must stop the queue to fix the problem or make modifications (see Table 3–15 and Table 3–16).

Table 3–15 Aborting and Requeuing Jobs

Possible Reason for Abort/Requeue	Command	Comments
Job contains unprintable characters and is using excessive amounts of paper.	`$ STOP/ABORT queue-name`	Aborts currently executing print job. Next job begins printing.
Current job is not as important as job recently queued. Requeue it to execute later.	`$ STOP/REQUEUE queue-name`	Current print job is aborted and moved to the end of the same queue. Next job begins printing.
Job is too large for this queue or its attributes are wrong. Move it to a more appropriate queue.	`$ STOP/REQUEUE=new-queue-name -` `_$ queue-name`	Current job is aborted and moved to the end of the new queue. Next job begins executing on the original queue.
Current job is not as important as more recently submitted job. Requeue current job to be executed later.	`$ STOP/ENTRY=job-number -` `_$ /REQUEUE queue-name`	Current job is stopped and placed at the end of the queue. The job with the next highest priority will begin executing. (The more important job should have been submitted with sufficiently higher priority to be the next job to execute.)

Table 3–16 Stopping Queues

Possible Reasons for Stopping a Queue	Command	Comments
Temporarily Suspend Queue Execution		

Paper is jammed in printer.
 	
Printer needs paper.
 	
Printer needs ribbon changed.
	`$ STOP/QUEUE queue-name`
	
	`$ STOP/QUEUE LPA0`
		Pauses queue. Suspends execution of current job. No new jobs can be initiated until queue is restarted with the `START/QUEUE` command.

Orderly Shutdown of a Queue		

Queue must be deleted or reassigned, or the jobs in it must be moved to another queue.	`$ STOP/QUEUE/NEXT queue-name`
	
	`$ STOP/QUEUE/NEXT LASER`
		Stops queue after the currently executing jobs are completed. Queue must be stopped before being deleted or reassigned, or before jobs can be moved.

Immediate Shutdown of a Queue		

You cannot stop or delete the current job.
 	
A previous queue control command did not produce the desired effect.
	`$ STOP/QUEUE/RESET queue-name`
	
	`$ STOP/QUEUE/RESET SYS$PRINT`
		Abruptly stops queue and stops all currently executing jobs. After this command, you have complete control over the queue and the jobs in it.

Close a Queue		

No more jobs should be entered into a queue.	`$ SET QUEUE/CLOSE queue-name`
	
	`$ SET QUEUE/CLOSE BANK_CHECKS`
		Prevents jobs from being entered in the queue through `PRINT` or `SUBMIT` commands or as a result of requeue operations. To allow jobs to be entered, use the `SET QUEUE/OPEN` command. When a queue is marked closed, jobs executing continue to execute and jobs already pending in the queue continue to be candidates for execution.

Table 3–16 (Cont.) Stopping Queues

Possible Reasons for Stopping a Queue	Command	Comments
Orderly Shutdown of All Queues		
All queue processing and execution must stop.	`$ STOP/QUEUE/MANAGER`	Performs an orderly shutdown of all queues. To restart the queues, you must first restart the queue manager by issuing the `START/QUEUE/MANAGER/RESTART` command, then restart all queues in the appropriate fashion, either manually or through a command procedure.

While a logical or generic queue is stopped, jobs are not sent to their associated execution queues. However, you can submit new jobs directly to the execution queues, or to the logical or generic queues where they will be held until the queues are restarted (see Table 3–17).

Table 3–17 Assigning and Deassigning Logical Queues

Operation	Commands and Examples

Change the assignment of the queue†	`$ STOP/QUEUE/NEXT logical-queue-name`
	`$ ASSIGN/QUEUE execution-queue-name -`
	`_$ logical-queue-name`
	
	`$ STOP/QUEUE/NEXT MEMOS`
	`$ ASSIGN/QUEUE LASER2 MEMOS`

Remove the assignment of the queue	`$ STOP/QUEUE/NEXT logical-queue-name`
	`$ DEASSIGN/QUEUE logical-queue-name`
	
	`$ STOP/QUEUE/NEXT SPECIAL_PRINTS`
	`$ DEASSIGN/QUEUE SPECIAL_PRINTS`

†You can assign a logical queue to only one execution queue at any time, but you can assign it to a different execution queue at any time.

To move all the jobs from one queue to another, stop the queue containing the jobs to be moved to prevent execution of any more jobs. If the queue is an execution queue, you should requeue the current job (see Table 3–18).

Table 3–18 Moving Jobs from One Queue to Another

Operation	Command Format	Example: LPA0 needs repair; LPB0 is still operational

Move all jobs from one queue to another.	`$ ASSIGN/MERGE -`
	`_$ destination-queue-name -`
	`_$ source-queue-name`
		`$ STOP/QUEUE LPA0`
		`$ STOP/QUEUE/REQUEUE=LPB0 -`
		`_$ LPA0`
		`$ ASSIGN/MERGE LPB0 LPA0`

After you move the jobs to another queue, you may want to delete the original queue. You must always stop the queue with the STOP/QUEUE/NEXT command before you can delete it (see Table 3–19).

Table 3–19 Deleting a Queue

Operation	Command Format	Example
Delete a queue.	`$ DELETE/QUEUE queue-name`	`$ STOP/QUEUE/NEXT LPA0` `$ DELETE/QUEUE LPA0`

You can stop print or batch execution queues with either the commands STOP/QUEUE or STOP/QUEUE/NEXT. If you use the STOP/QUEUE/NEXT command, the system completes the current jobs before stopping the queue. Therefore, when you start the queue again, the system begins executing the next available job.

If you use the STOP/QUEUE command, the system suspends the execution of the current jobs. You can delete suspended jobs while the queue is stopped or when you restart the queue (see Table 3–20). If you do not delete the job, the system will continue to execute it when you restart the queue.

Table 3–20 Deleting a Job in a Queue

Operation	Command Format	Example
Delete a job in a queue.	`$ DELETE/ENTRY=job-number`	`$ DELETE/ENTRY=715`
Delete the current job when restarting a print execution queue.	`$ START/QUEUE/NEXT queue-name`	`$ START/QUEUE/NEXT LPA0`

For more information on controlling print queues, read the command descriptions and qualifier descriptions for the STOP/QUEUE, ASSIGN/QUEUE, ASSIGN/MERGE, DELETE/QUEUE, and START/QUEUE commands in the *VMS DCL Dictionary*.

3.10.2 Managing Printer Forms and Characteristics

A print queue can be specified as having a particular **form** mounted on the associated printer, or possessing one or more special **characteristics**. Forms and characteristics are used to further qualify which jobs can execute on particular printers. System managers define forms and characteristics and associate them with print queues; users indicate the need for a particular form or set of characteristics with the commands PRINT and SET ENTRY.

Typically, forms are defined and associated with print queues to make it easier to execute jobs requiring different paper stocks or sizes. Characteristics are quite similar, but are usually used to separate the execution of jobs with other nonpaper specifications, such as different colored ribbons, and so on. Some users define special characteristics for a plotter associated with a print queue, thereby allowing for different pen colors and tips. In general, most system managers take advantage of forms definitions, but seldom use the characteristics capability.

3.10.2.1 Creating Forms and Characteristics

Use the DEFINE command to create forms and characteristics:

```
$ DEFINE/FORM  form-name  form-number  [/qualifiers]
$ DEFINE/FORM WIDE 3 /WIDTH=132 /LENGTH=60 /STOCK=GREEN_BAR

$ DEFINE/CHARACTERISTIC  characteristic-name  characteristic-number
$ DEFINE/CHARACTERISTIC  RED_INK  74
```

In each case, the name and number to be used is arbitrary, but each must be unique on the system. The name can have from 1 to 31 characters, including uppercase and lowercase letters, digits, the dollar sign, and the underscore; however, the name must contain at least one nonnumeric character. Form numbers can range from 0 to 999, and characteristic numbers from 0 to 127.

There is no special meaning to the names and numbers used; it is up to the system manager to keep track of which names and numbers are assigned on a system and what they mean. It is important that the system manager properly inform system users of these arbitrary names, their intended uses, and which printers are associated with which forms and characteristics. Use the /DESCRIPTION qualifier to the DEFINE command to document the intended use of the form or characteristic for users (see Table 3–21). The SHOW/QUEUE/FORM and SHOW/QUEUE/CHARACTERISTICS commands allow users to see which forms and characteristics are defined on the system (see Example 3–6).

A form definition can consist of one or more attributes, as illustrated in Table 3–21. Note that the queue manager must be running the previous issuance of a START/QUEUE/MANAGER command before you can define forms or characteristics.

Table 3–21 Qualifiers for the DEFINE/FORM Command

Qualifier	Function
/DESCRIPTION=string	Assigns a description to the form. Can be as long as 255 characters, with the default being the name of the form.
/LENGTH=n	Specifies the physical form length in lines. The default is 66 lines.
/MARGIN=(option[,...])	Specifies one or more of the following four margin options:
	LEFT=n — The number of columns to be left blank between the left-most printing position and the actual print image area.
	RIGHT=n — The number of columns to be left blank between the /WIDTH setting and the actual print image area.
	TOP=n — The number of blank lines to leave between the top of the physical page of paper and the start of the print image.
	BOTTOM=n — The number of blank lines to leave between the end of the print image on a page and the end of the physical page of paper.
/PAGE_SETUP=(module[,...])	Specifies one or more modules that set up the printer before every page.
/SETUP=(module[,...])	Specifies one or more modules that set up the printer before a file is printed.
/SHEET_FEED	Tells the print symbiont associated with the queue to pause at the end of every physical page so that a new piece of paper can be inserted.
/STOCK=string	Names the paper stock associated with the form. The string can be from 1 to 31 characters, including all letters, digits, the dollar sign and the underscore. The default for string is the form name.
/TRUNCATE	Tells the printer to discard all characters that would exceed either the /WIDTH value or the /MARGIN=RIGHT value. You cannot specify both /TRUNCATE and /WRAP for the same form. Specify /NOTRUNCATE and /NOWRAP for special graphics output devices.

Table 3–21 (Cont.) Qualifiers for the DEFINE/FORM Command

Qualifier	Function
/WRAP	Tells the printer to print all characters exceeding either the /WIDTH value or the /MARGIN=RIGHT value on the next line.
/WIDTH=n	Specifies the physical width of the paper in terms of columns or character positions.

```
$
$ SHOW QUEUE/FORM
Form name                              Number   Description
---------                              ------   -----------
ALLIN1 (stock=DEFAULT)                     99   ALLIN1 Memos
BORDER (stock=DEFAULT)                      4   Border Outline
DEFAULT                                     0   System-defined default
FRAME (stock=DEFAULT)                       3   Overhead Frame
TEMP                                        6   Special temporary
WPSPLUS (stock=DEFAULT)                  1101   WPSPLUS Documentation
$
$
$ SHOW QUEUE/CHARACTERISTICS
Characteristic name                    Number
-------------------                    ------
PEN_010                                     5
PEN_020                                     6
PEN_050                                     7
RED_INK                                    74
$
```

Example 3–6 Displaying Queue Forms and Characteristics

3.10.2.2 Using Forms and Characteristics with Printer Queues

Once a form or characteristic is defined, you can use the /FORM_MOUNTED qualifier with either the INITIALIZE/QUEUE, START/QUEUE, or SET QUEUE commands (see Table 3–22).

More detailed information about defining forms and characteristics can be found in the *Guide to Maintaining a VMS System*.

Table 3–22 Defining Printer Forms and Characteristics

Command†	Comments
`$ INITIALIZE/QUEUE -` `_$ /CHARACTERISTICS=(# or name,[,...])†`	Characteristics of jobs must match or be a subset of queue characteristics for the job to execute. The `DEFINE/CHARACTERISTICS` command sets up a correspondence between the characteristic name and number. Note that a characteristic must already be defined before it can be used with a queue.

`$ DEFINE/CHAR REDINK 2`
`$ INITIALIZE/QUEUE/START/CHAR=REDINK PLOTTER`
`$ PRINT/CHAR=REDINK/QUEUE=PLOTTER FILE.DAT`
	Set up the characteristic name REDINK to correspond to the number 2. Specify the REDINK characteristic as a characteristic of the PLOTTER queue. Only jobs sent to this queue that specify the REDINK characteristic by name (or number) will be executed from this queue. Other jobs will be placed on hold.

`$ INITIALIZE/QUEUE/FORM=(# or name,[,...])`	Limits jobs to those whose form name (or number) matches that of the queue. Note that a form must already be defined before it can be used with a queue.
`$ DEFINE/FORM WIDE 2` `$ INITIALIZE/QUEUE/START/FORM=WIDE LPA0`	Define a new form called WIDE and initialize (and start) the LPA0 queue with this new form.
`$ PRINT/FORM=WIDE/QUEUE=LPA0 FILE.DAT`	Submit a request to print FILE.DAT on the LPA0 queue with the WIDE form.

†It is generally considered a bad practice to specify forms or characteristics by their numbers. Use the form or characteristic name to better document the intended action and to keep from inadvertently specifying the wrong form or characteristic.

3.10.3 Handling Print Queue Problems

The output for a suspended and continued print job may not be correct due to a printer or paper malfunction. The paper in the printer may have jammed, destroying the pages most recently output. Or, the paper may have run out and the operator does not know how to load new paper correctly. In these and other situations, you must be able to reprint all or part of a job. You may also need to print a few pages of text that are not part of the job to align the paper.

In many cases, you can see how much of the job completed successfully, and how much will have to be redone. To reprint the entire job, requeue it (STOP/REQUEUE). This is an effective use of your time and the printer resource for small jobs, for jobs that have just started, or when you cannot tell how many pages of the job were destroyed (for example, when a printer jams but continues to print information, eventually ripping the paper, instead of going off-line).

For jobs where you can determine how many pages to reprint, or for large jobs with occasional lines of text that are unique in the file, use the commands shown in Table 3–23 to specify where in the job the system should begin to print when you start the queue.

Before you restart the queue, you may need to align the paper. If you include the /ALIGN qualifier with the START/QUEUE command, one page of the job prints (by default) and the queue pauses. After manually adjusting the paper, restart the queue. As Table 3–24 shows, you can use one of the qualifiers from Table 3–23 to position the job when you restart the queue.

Table 3–23 Positioning a Print Job

START/QUEUE Qualifier	Comments
/BACKWARD=n	File is backspaced n pages before printing is resumed.
/FORWARD=n	File is forward spaced n pages before printing is resumed.
/SEARCH=string	Resumes printing with page containing string. (Search direction is forward. Other qualifiers processed first.)
/TOP_OF_FILE	Printing begins at top of interrupted file (not top of job).

Table 3–24 Aligning Printer Paper

Command Format	Comments
`$ START/QUEUE/ALIGN queue-name`	One page of the job is printed. The queue stops. Adjust the paper and restart the queue.
`$ START/QUEUE/BACKWARD=2/ALIGN=2 LPA0` `$ START/QUEUE LPA0`	You can back up several pages before beginning the reprint. In this example, the symbiont backs up two pages in the job, then prints two alignment pages and stops. The user adjusts the paper and restarts the queue. The system begins printing the next page in the job.

3.11 Laboratory Exercises

1. Send two separate print jobs to your terminal execution queue, holding them until you release them.
2. Release the first job, then abort it.
3. Display the contents of the queue immediately after aborting your job.
4. Stop the queue using the /RESET qualifier. You now have complete control over the queue.
5. Release the next job being held in the queue.
6. Display the contents of the queue and note the status of the jobs.
7. Start the queue and observe the output on your terminal.
8. Stop and delete your terminal output queue and your batch queue, MULTIBAT, that you created in the previous laboratory exercise.
9. Set the output terminal back to its original characteristics. (You will need OPER and LOG_IO privileges to do this.)

3.12 Solutions to Laboratory Exercises

Substitute your file names, entry numbers, and queue names in the commands listed below.

1.
```
$ PRINT/QUEUE=MYQUEUE/HOLD FILE1.TXT
$ PRINT/QUEUE=MYQUEUE/HOLD FILE2.TXT
```

2.
```
$ SET ENTRY 369/RELEASE
$ STOP/ABORT MYQUEUE
```

3.
```
$ SHOW QUEUE MYQUEUE
```

   Note the status of the jobs on the queue. Also note that trailer pages, if specified, will be printed for a job that was aborted.

4.
```
$ STOP/QUEUE/RESET MYQUEUE
```

5.
```
$ SET ENTRY 369/RELEASE
```

   Notice that the job did not print, as the queue is stopped.

6.
```
$ SHOW QUEUE MYQUEUE
```

7.
```
$ START/QUEUE MYQUEUE
```

8.
```
$ STOP/QUEUE/NEXT MYQUEUE
$ DELETE/QUEUE MYQUEUE
$ STOP/QUEUE/NEXT MULTIBAT
$ DELETE/QUEUE MULTIBAT
```

9.
```
$ SET DEVICE/NOSPOOL TTC5:
$ SET TERMINAL/PERMANENT/TYPE_AHEAD/BROADCAST TTC5:
```

   Substitute the device name of your output terminal for TTC5.

3.13 How the VMS System Handles Batch Jobs

When you execute a command procedure in your interactive process, you cannot enter any other commands until the procedure completes. This may be acceptable if you do not mind waiting or if there is another terminal available for you to use. However, most users have access to only one terminal at a time.

To avoid unnecessary waiting and to make the best use of terminal resources, you can create a **command procedure** that contains commands and data input lines. A command procedure can then be submitted to a **batch queue** for "hands free" execution. A command procedure that executes in a batch queue is called a **batch job**.

One of the responsibilities of the system manager is to assess the needs of the user community for batch and print facilities; and to create queues with attributes and characteristics that match those needs. Once a batch queue exists, you can use the SUBMIT command to send your batch job to it.

> NOTE
> If you do not create batch queues, the VMS operating system will not execute batch jobs on your system.

At some sites, users create **cards** containing their command procedure. They enter one line of the procedure on each card. After adding job cards to the beginning and end of the card deck, they place them in a card reader. For jobs submitted through a card reader, the VMS operating system creates a file on a disk and writes each line of the job to the file. Then it submits the file to a queue (by default, the SYS$BATCH queue). This is called **input spooling**. You do not have to set up input spooling. The VMS operating system does it automatically when it reads a job from a card reader (see Figure 3-8).

> NOTE
> The VMS operating system keeps all of the batch queue information as well as print queue information in SYS$SYSTEM:JBCSYSQUE.DAT (See the discussion of this file in an earlier section of this module.)

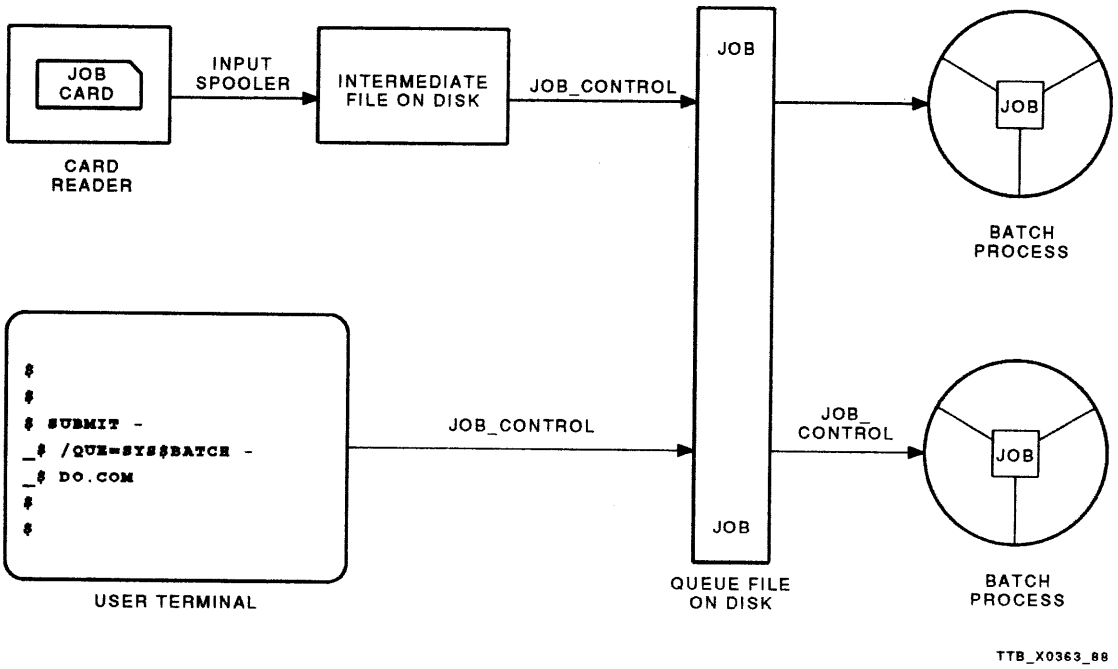


Figure 3–8 JOB_CONTROL Process Handles All Batch Jobs

For each job, the JOB_CONTROL process creates a batch process to execute the job. It assigns a name to the batch process composed of the word BATCH and the job identification number. When a batch process has been created for a batch job, the job is listed as a currently executing job in the queue. When you list processes with the SHOW SYSTEM command, a B appears in the right column for each batch job (see Example 3–7).

```
$
$ SHOW SYSTEM
VAX/VMS 5.2  on node BROWNY  19-APR-1989 16:55:43.02   Uptime  10 03:36:27
  Pid    Process Name    State  Pri      I/O       CPU        Page flts Ph.Mem
20200021 SWAPPER         HIB     16        0    0 00:00:21.96         0      0
202002A2 Chocoholic      HIB      9     4000    0 00:01:00.00      6658    328
20200263 DUFFY           LEF      4      328    0 00:00:10.49      1057    299
20200185 VPA_DC          HIB     15     4623    0 00:13:00.43      1374   1333
20200027 ERRFMT          HIB      8     9995    0 00:02:59.54        82    118
20200028 CACHE_SERVER    HIB     16      152    0 00:00:00.75        62     93
20200029 CLUSTER_SERVER  HIB      8       39    0 00:00:02.20       151    314
2020002A OPCOM           HIB      8     4321    0 00:02:13.76       645    211
❶202001B8 BATCH_103      CUR      3       80    0 00:00:02.93       588    259  B ❷
2020002B AUDIT_SERVER    HIB     10       54    0 00:00:54.84      1300    223
❸2020002C JOB_CONTROL    HIB      9     1875    0 00:00:14.57       207    412
2020002D CONFIGURE       HIB     10      122    0 00:00:12.96       111    159
2020002E SMISERVER       HIB      9      104    0 00:00:03.07       406    437
20200251 NETACP          HIB     10     1493    0 01:24:58.28   2321834   3500
❹202001A3 _CRA0:         LEF      4     2501    0 00:00:58.87        50     41
20200112 EVL             HIB      5     1390    0 00:00:39.21     49642     38  N
202001B3 REMACP          HIB      9       59    0 00:00:00.56        80     50
20200075 WOODS           LEF      4    14551    0 00:09:09.52     37853    170
20200079 _RTA1:          HIB      6    22780    0 00:20:36.65     10459   4096
2020029A SPM_CAPACITY    LEF     24     1975    0 00:00:40.75       217    170
$
```

Example 3–7 JOB_CONTROL, Input Symbiont and Batch Job Processes

Notes on Example 3–7:

❶ Batch process.

❷ The letter B indicates that this process is executing a batch job.

❸ The JOB_CONTROL process.

❹ The input symbiont process name is the same as the device name of the card reader from which the job is read.

3.13.1 Batch Job Scheduling

The system manager limits the total number of processes (interactive, batch, network, and other types) that can execute concurrently on the system. Typically, when the manager creates a batch queue, he or she limits the number of jobs that can run concurrently from the queue as well. Because of these limitations, the VMS operating system must schedule batch jobs to run. It schedules them according to their priority and time of submission. It executes the job with the highest priority first. Within the same queue priority, jobs are executed in order according to their time of submission - earliest jobs first (see Figure 3-9 and Example 3-8).

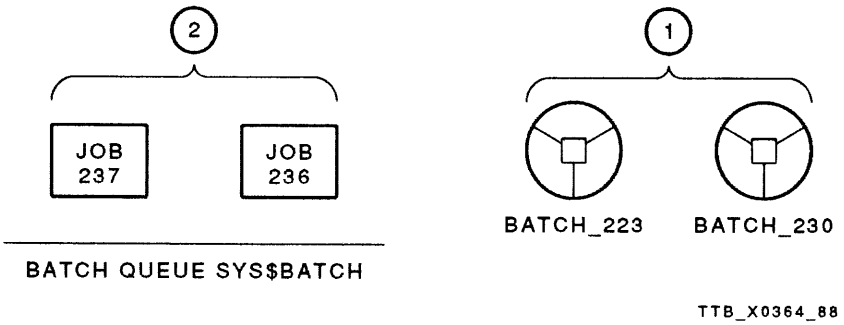


Figure 3–9 Jobs on a Batch Queue

```
$
$ SHOW QUEUE/FULL SYS$BATCH
Batch queue SYS$BATCH
/BASE_PRIORITY=3 /JOB_LIMIT=2 /OWNER=[SYSTEM] /PROTECTION=(S:E,O:D,G:R,W:W)

  Jobname          Username     Entry  Status
  -------          --------     -----  ------
  ACTION           JONES          223  Executing
❶   Submitted 13-DEC-1987 12:44 /PRIORITY=100
      _DJA0:[JONES]ACTION.COM;2 (executing)

  MATH             JONES          230  Executing
    Submitted 13-DEC-1987 12:57 /PRIORITY=100
      _DJA0:[JONES]MATH.COM;1 (executing)

❷ COMPUTE          JONES          237  Pending
    Submitted 13-DEC-1987 13:41 /PRIORITY=120
      _DJA0:[JONES]COMPUTE.COM;7 (pending)

❸ ACTION           JONES          236  Pending
    Submitted 13-DEC-1987 13:10 /PRIORITY=100
      _DJA0:[JONES]ACTION.COM;3 (pending)

$
$
```

Example 3–8 Current and Pending Jobs on a Batch Queue

Notes on Figure 3–9 and Example 3–8:

❶ Jobs 223 and 230 are currently executing. The JOB_LIMIT attribute of the queue has a value of two, limiting the number of concurrent batch processes from this queue to two. Because the number of concurrent batch processes running from SYS$BATCH equals its job limit, Job 236 and Job 237 must wait to execute. Their status is **pending**. The JOB_CONTROL process examines the parameters of the pending jobs to determine which job it will execute next.

❷ Since Job 237 has a higher queue priority than Job 236, (see note 2 of Example 3–2) it will be executed next.

NOTE
Although the queue priority of a batch job helps to determine when it will be scheduled, the queue priority does not affect the base priority of the batch process. Base priority is defined by:

- The value of the BASE_PRIORITY queue attribute; in this case, three (3).
- The value of the system parameter DEFPRI. The system uses this value if you do not set the BASE_PRIORITY attribute for a queue.

❸ Finally, Job 236 executes. However, if a user submits another job before Job 236 begins executing, the JOB_CONTROL process compares the parameters of the new job with those of Job 236 to determine which it will execute next.

3.14 Batch Queue Operations

Operations on batch queues are very similar to those performed on print queues. Almost everything presented on print queues pertains to batch queues, except such print-specific situations as separation pages, printer forms, and job positioning commands. Two functional areas that differ slightly from print queues are in creating and stopping batch queues. These topics are discussed in separate sections below. Batch queues can be specified as having special characteristics, but this feature is seldom used on most systems.

3.14.1 Types of Batch Queues

There are two types of batch queues - execution and generic. Note that logical queues are not available for batch queues.

3.14.2 Creating Batch Queues

Normally, you create the SYS$BATCH queue (the default queue for the SUBMIT command and spooled jobs from card readers) with default parameters by using the following command:

```
$ INITIALIZE/QUEUE/BATCH/START SYS$BATCH
```

Note the /BATCH qualifier in the command. This qualifier is **required** for the VMS operating system to distinguish between the creation of print and batch queues. If you were to leave out the /BATCH qualifier in the above command, the VMS operating system would attempt to create a print queue called SYS$BATCH. See Table 3–25 for sample batch queue names and parameter values. See Table 3–26 for a list of qualifiers used with INITIALIZE/QUEUE.

Table 3–25 Batch Queue Names and Parameter Values

Suggested Name and Purpose of the Queue	Parameters and Comments
GROUP360 — used by those whose group UIC is 360 because they are working on a high-priority project.	Set the owner UIC of the queue to [360,000]. Give GROUP users READ and WRITE access only. Give WORLD users no access. Set larger working set extent, CPU limits, and priority. Possibly increase the job limit.
FASTQUE — used by all who need a job done quickly.	Set the base priority at 5. Use default queue protection or possibly restrict use to a certain group, UIC, or ACL. Set job limit at 2, but limit the maximum CPU to a low value to keep the queue from taking over the system. Optionally use higher working set limits.
ZOOMQUE — very fast queue that you only start after hours or during lunch.	Set the base priority at 6 or higher, but definitely set a low CPU limit. Set job limit to 2 and high working set limits. Setting a batch queue's default BASE_PRIORITY to a value higher than the normal interactive value is considered dangerous to system response time. Carefully monitor this batch queue to avoid system degradation.
CADCAM — to be used by large, compute-intensive applications, such as engineering, manufacturing, or modeling programs.	Set the priority at 3. (DO NOT set to 0 or 1, as jobs may get very little CPU time and finish too slowly.) Increase working set limits to large values. Set CPU limit to very large value, or infinite. Optionally protect the queue for access by only specific user groups.

Table 3–26 Qualifiers to INITIALIZE/QUEUE for Batch Queues

Qualifiers	Examples	Function
/JOB_LIMIT	`$ INITIALIZE/QUEUE/BATCH -` `_$ /JOB_LIM=2 FASTBAT`	Sets a limit on the number of batch processes that can run concurrently from one batch queue. The default is 1. The higher the job limit, the greater the number of processees that will execute from the queue concurrently, and the greater the drain on the system resources.
/BASE_ PRIORITY	`$ INITIALIZE/QUEUE/BATCH -` `_$ /BASE=5 FASTBAT`	Defines the base priority of a batch process. The system parameter DEFPRI sets the default. The higher the priority, the sooner it is scheduled to run. If many batch processes have priorities as high as or higher than interactive process priorities (default 4), they can degrade the performance for interactive users and the whole system.
/PROTECTION /OWNER_ UIC	`$ INITIALIZE/QUEUE/BATCH -` `_$ /OWN=[ENG,PROJ5]`	Limits access to a queue.

/WSDEFAULT
/WSQUOTA
/WSEXTENT
/DISABLE_ SWAPPING
	`$ INITIALIZE/QUEUE/BATCH -`
	`_$ /WSDEF=500 -`
	`_$ /WSQUOT=800 -`
	`_$ /WSEXT=2000 -`
	`_$ /DISABLE_SWAP -`
	`_$ FASTBAT`
		Defines memory management parameters for the batch process (limits working set size and adjustment allowed; sets swap or noswap). UAF values are the default. If you set high values for these and disable swapping of processes for the queue, you can use so much memory that system performance is degraded (because paging and swapping increases for interactive jobs).

Table 3–26 (Cont.) Qualifiers to INITIALIZE/QUEUE for Batch Queues

Qualifiers	Examples	Function
/CPUMAXIMUM	$ INITIALIZE/QUEUE/BATCH - _$ /CPUMAX=INFINITE - _$ BIGJOBBAT	Sets the maximum CPU limit to be assigned to a batch process from the queue. The default maximum is the CPU limit in the owner's UAF record. The maximum (however established) is the default assigned to batch processes unless it is overridden with the SUBMIT command or the /CPUDEFAULT initialization qualifier, both of which can assign a smaller default. If you allow large CPU limits for an untested job, the job could have an infinite loop and degrade system performance. To improve response time for interactive users, set small CPU limits in user UAF records and large limits in a low priority batch queue, forcing users to run lengthy programs as batch jobs. You might also assign larger working sets to this type of queue.
/CPUDEFAULT	$ INITIALIZE/QUEUE/BATCH - _$ /CPUDEF=00:03:00 - _$ 3MINBAT	Sets default CPU limit assigned to batch processes from this queue. Otherwise, the default is the CPU limit in a user's UAF record or the value of /CPUMAXIMUM for the queue.

3.14.3 Stopping Batch Queues

When you stop a batch queue with the STOP/QUEUE command, the system suspends execution of the current jobs, but when you stop a batch queue with the STOP/QUEUE/NEXT command, the system completes the current jobs before stopping the queue. (See the display from the SHOW SYSTEM/BATCH command in Example 3–9.) While the current jobs are being completed, the state of the queue is **stop pending**, and no new jobs can be initiated from it (see Example 3–9).

```
$
❶ $ SUBMIT ACTION
Job ACTION (queue SYS$BATCH, entry 911) started on SYS$BATCH
$
$ STOP/QUEUE SYS$BATCH
$
$ SHOW QUEUE/ALL SYS$BATCH
❷ Batch queue SYS$BATCH, paused

  Jobname          Username     Entry  Status
  -------          --------     -----  ------
  ACTION           JONES          911  Executing
$
$ SHOW SYSTEM/BATCH
VAX/VMS V5.0  on node BROWNY 13-DEC-1987 13:30:24.38   Uptime   12 22:30:54
  Pid    Process Name    State  Pri      I/O       CPU        Page flts Ph.Mem
❸ 202003C5 BATCH_911       SUSP     4       25   0 00:00:00.71       143    171  B
$
$
❹ $ START/QUEUE SYS$BATCH
$
❺ $ STOP/QUEUE/NEXT SYS$BATCH
$
$ SHOW QUEUE/ALL SYS$BATCH
❻ Batch queue SYS$BATCH, stop pending

  Jobname          Username     Entry  Status
  -------          --------     -----  ------
  ACTION           JONES          911  Executing
$
$ SHOW SYSTEM/BATCH
VAX/VMS V5.0  on node BROWNY 13-DEC-1987 13:31:46.62   Uptime   12 22:31:33
  Pid    Process Name    State  Pri      I/O       CPU        Page flts Ph.Mem
❼ 202003C5 BATCH_911       LEF      4       25   0 00:00:00.71       143    171  B
$
$ SHOW QUEUE/ALL SYS$BATCH
❽ Batch queue SYS$BATCH, stopped
$
❾ $ SHOW SYSTEM/BATCH
$
$
```

Example 3–9 Stopping Batch Queues

Notes on Example 3–9:

❶ The command procedure ACTION.COM is submitted and placed into the default system batch queue, SYS$BATCH since an explicit /QUEUE qualifier was not specified in the command. The queue manager assigned the entry number of 911 to the queue request.

❷ Examining the queue shows its state is **paused**, but the state of the batch job is **Executing**.

❸ Note that the system indicates the executing batch job as being in a **suspended** (SUSP) state.

❹ Start the SYS$BATCH batch queue running again.

❺ Inform the queue manager to stop the SYS$BATCH queue after the current batch job has finished executing.

❻ The current batch job continues executing, but the SYS$BATCH queue state indicates a pending stop on the queue. New jobs can be entered into the queue, but they will remain in a **pending** state until the queue is restarted.

❼ Note that the system indicates the current batch job is running and is in a Local Event Flag (LEF) wait state.

❽ Now we see the SYS$BATCH queue in a **stopped** state and that there are no jobs executing or awaiting execution.

❾ The system indicates no batch jobs are currently executing.

For more information on the commands used to set up batch queues, read their command descriptions in the *VMS DCL Dictionary*, or consult the *VMS System Manager's Manual*.

3.15 Laboratory Exercises

1. Find out what batch queues are already on the system.
2. Create and start a batch queue with a job limit of two and a priority of one. Make sure the queue name is different from any existing queue names.

3.16 Solutions to Laboratory Exercises

1. `$ SHOW QUEUE /BATCH`
2. `$ INITIALIZE /QUEUE /BATCH /START /JOB_LIMIT=2 -`
   `_$ /BASE_PRIORITY=1 SMITH_BATCH`

3.17 Restricting Access and Control of Queues

There are two ways a system manager can control access to queues: UIC-based protection and ACL-based protection. Using ACLs to control access to queues is a new feature available with VMS Version 5.

3.17.1 UIC-Based Queue Protection

The format of the queue protection mask is similar to that of the file protection mask. The system uses the mask in conjunction with the owner UIC of the queue to allow or reject user access to the queue. Users attempting to gain access to a queue are categorized as System, Owner, Group, or World users according to their UIC and the owner UIC of the queue (see Table 3–27). After categorizing a user, the system allows the user to gain access to the queue according to the access listed for his or her category (see Table 3-28).

Table 3–27 User Categories

Category	Comments
System	User has a system UIC
Owner	User UIC matches queue's owner UIC
Group	Group UIC number of user matches group UIC number of queue owner
World	User UIC does not match queue's owner UIC

The default protection code is (System:E, Owner:D, Group:R, World:W). As with file protection codes, each category of user has all the privileges assigned to less privileged categories. For example, in the default code, a user in the Owner category has Delete, Read, and Write access to the queue on jobs in the queue.

Table 3–28 Access to Queues

Access Codes	Access Allowed
Read	Can list the attributes of a job
Delete	Can modify or delete any job, or delete the queue
Write	Can submit jobs to the queue
Execute	Can affect the queue and/or all jobs in the queue (users with OPER privilege have E access to all queues)

3.17.2 ACL-Based Queue Protection

In addition to UIC-based protection, you can provide more refined queue access and control by using access control lists (ACLs). You may often have a group of users who want total control over the queue associated with a privately owned printer. Using ACLs gives you the ability to completely control the use and management of a queue among an arbitrary set of users.

For example, assume some users have purchased an LN03 printer for their private use. Most of the users are part of a manufacturing group, but there are also users within an engineering group who will require access to the printer. The users also want to explicitly exclude all other users on the system from accessing the printer. Furthermore, one of the manufacturing users will serve as a sort of "local operator" responsible for local management of the printer and its queue.

To set up a privately controlled queue for the LN03, you must perform the following steps:

1. Create one or more ACL **identifiers** for use in associating the target user group and the printer queue.
2. Grant the identifiers to the target user group.
3. Set the printer queue for appropriate access by the target user group.

To illustrate this process, assume the target user group consists of three manufacturing users (MFG_USER1, MFG_USER2, and MFG_USER3) and two engineering users (ENG_USER1 and ENG_USER2). Also, it has been determined that MFG_USER1 will serve as the "local operator" and as a liaison with the system manager. That is, all members of the target group can submit jobs to the queue, but only MFG_USER1 can perform queue management operations (in addition to the system manager, operators, and any users with the OPER privilege). The steps needed to set up the ACL environment and the printer queue are shown in Table 3–29.

Table 3–29 Preparing a Privately Controlled Printer

Operation	Command	Comments

Create identifiers	`$ SET DEFAULT SYS$SYSTEM`
	`$ RUN AUTHORIZE`
	`UAF> ADD/ID MFG_PRT`
	`UAF> ADD/ID MFG_PRT_OPER`
		Create the MFG_PRT identifier for users allowed to submit jobs to the print queue, and the MFG_PRT_OPER for users responsible for managing the queue.

Grant identifiers to appropriate users	`UAF> GRANT/ID MFG_PRT MFG_USER1`
	`UAF> GRANT/ID MFG_PRT MFG_USER2`
	`UAF> GRANT/ID MFG_PRT MFG_USER3`
	`UAF> GRANT/ID MFG_PRT ENG_USER1`
	`UAF> GRANT/ID MFG_PRT ENG_USER2`
	`UAF> GRANT/ID MFG_PRT_OPER MFG_USER1`
	`UAF> EXIT`
		Associate the MFG_PRT identifier with the five users, and give MFG_USER1 the additional MFG_PRT_OPER identifier.

Set queue protection†	`$ SET QUEUE/PROTECTION=(S,O,G,W) MFG_LN03`
	
	`$ SET ACL/ACL=(IDENTIFIER=MFG_PRT_OPER, -`
	`_$ ACCESS=READ+WRITE+DELETE+EXECUTE) -`
	`_$ /ACL=(IDENTIFIER=MFG_PRT, -`
	`_$ ACCESS=READ+WRITE+DELETE) -`
	`_$ /OBJECT=QUEUE MFG_LN03`
		First, set the printer queue for NO ACCESS by any user by specifying a UIC mask with no access parameters. Next, associate the two previously created identifiers with the MFG_LN03 printer queue. The EXECUTE access type gives the MFG_USER1 special queue control capabilities, such as starting and stopping the queue and modifying or deleting jobs.

†This example assumes the MFG_LN03 printer queue has already been created using one of the previously described queue creation procedures.

Additional users can be given access to the MFG_LN03 queue by granting them the MFG_PRT identifier using the Authorize utility. ACLs are discussed further in Module 8. For more information about ACLs and applying ACLs to queues, see the *Guide to VMS System Security*, the *VMS Access Control List Editor Manual*, and the description of the SET ACL command in the *VMS DCL Dictionary*.

3.18 Overview of Queue Commands

Most of the commands in Table 3–30 require the OPER privilege to execute. A few queue-related operations require other privileges, such as LOGIO or ownership rights, to properly execute. A user is able to modify or delete his own queue jobs; however, OPER privilege or Execute (E) access is required to modify other users' queue jobs.

Table 3–30 Queue-Related DCL Commands

DCL Command	Command Description
Creating/Controlling/Deleting Queues	
`INITIALIZE/QUEUE`	Creates and initializes a queue
`ASSIGN/QUEUE`	Assigns a queue to a device
`ASSIGN/MERGE`	Moves jobs from one queue to another
`START/QUEUE`	Starts or restarts a queue
`STOP/QUEUE`	Controls queue or current entry in it
`DEASSIGN/QUEUE`	Deassigns a queue from a device
`DELETE/QUEUE`	Deletes a queue and all its entries

Table 3–30 (Cont.) Queue-Related DCL Commands

DCL Command	Command Description
Setting Queue Attributes	
SET QUEUE	Sets various queue parameters
SET ACL/OBJECT=QUEUE	Sets the access rights of a queue
EDIT/ACL/OBJECT=QUEUE	Edits the access rights of a queue
DEFINE/FORM	Sets the attributes of a form
DEFINE/CHARACTERISTIC	Defines a queue characteristic
Setting Job Attributes	
PRINT	Places an entry in a print queue
SUBMIT	Places an entry in a batch queue
SET ENTRY	Changes the status of a pending entry in a queue
DELETE/ENTRY	Deletes a pending entry from a queue
Monitoring Queue and Entry Status	
SHOW QUEUE	Displays status of entries in a queue
SHOW ENTRY	Displays status of an individual job entry

MANAGING DISK AND TAPE VOLUMES

4.1 Introduction

System managers must decide how to allocate the disk and tape devices on their systems. They normally allocate some devices for public use and some for private use.

Public volumes are disk volumes that all system users can access. They contain system code, system data, and files useful to all users. Private volumes are disk and tape volumes that only a single user or group of users can access. Private volumes typically contain user-specific or group-specific code or data.

The system manager coordinates disk and tape management with the system users and operators. The manager decides whether a volume should be public or private. The manager loads, initializes, mounts, unloads, and maintains public volumes. Users (possibly with the help of the manager or an operator) do the same for private volumes.

Maintenance of public volumes includes:

- Allocating disk space to users
- Tracking the use of disk space (with the SYSMAN utility)
- Scheduling backups
- Installing system software and layered product software on the volume
- Monitoring device errors (covered in more detail in Module 8)

The manager also helps maintain private volumes by:

- Scheduling a disk or tape device for private use
- Training users to use the equipment properly (loading volumes, starting the drive, using the volume and unloading it when the job is done)
- Responding to REQUEST/REPLY commands in system environments where users may not gain access to the equipment

4.2 Objectives

To share limited disk resources among users and processes, a system manager should be able to:

- Describe the uses of public disk and tape volumes
- Manage volumes, including:
  - Prepare volumes for use
  - Obtain and modify volume information
  - Use public volumes properly
  - Control the allocation of public volumes
  - Create a volume set
  - Maintain public and private volumes
  - Use the Backup utility to back up and restore information on volumes
- Use the Verify utility to check the validity of the file structure on a disk.
- Transfer files between the VMS system and PDP-11 systems.

4.3 Resources

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

1. *Guide to Maintaining a VMS System*
2. *VMS DCL Dictionary*
3. *Guide to Setting Up a VMS System*
4. *VMS Exchange Utility Manual*
5. *VMS Backup Utility Manual*
6. *VMS Bad Block Locator Utility Manual*
7. *VMS Analyze/Disk_Structure Utility Manual*
8. *VMS Mount Utility Manual*

The following documents contain information related to the material presented in this module:

1. *VMS Installation and Operations* manual for your specific VAX processor
2. *Guide to VMS Files and Devices*
3. *Guide to Using VMS Command Procedures*
4. *VMS I/O User's Reference Manual: Part I*
5. *VMS I/O User's Reference Manual: Part II*
6. *RMS-11 User's Guide*

To complete the Laboratory Exercises in this module, you must have access to at least one of the following:

- A blank tape volume and a tape device
- A blank disk volume and a corresponding disk device

4.4 Using Private Disk and Tape Volumes

Private disk and tape volumes are volumes that a user, or a group of users, owns exclusively. They have three main purposes:

- Preserving files
- Transferring files from one system to another
- Providing a private environment in which to work

4.4.1 Preserving Files

Most of the files you create are stored on your default disk, which you share with other system users. This device is called a **public disk**.

Although the protection you establish for your files is usually sufficient to guard them against inadvertent deletion, they are still vulnerable to the activities of more privileged users and the system manager. For this reason, most users make copies of their most important material on "backup" volumes, such as other disk packs or tape reels.

Typically, users "own" their backup volumes. A volume you own is called a **private volume;** it has an owner user identification code (UIC) identical to your own.

4.4.2 Transferring Files

You may find it necessary to transfer files between systems that are not connected by a communications link. In such circumstances, you must be able to physically move your files from one location to another. You can do this by copying your files to a portable volume, such as a tape reel or disk pack, and carrying that volume to the other system.

4.4.3 Providing a Private Environment

On occasion, you may want to perform your work on a device to which no one else has access. By creating a private volume and mounting it on a device assigned exclusively to your process, you can work without interference from other users.

4.5 Using Public Volumes

Public volumes are not owned by any user or group exclusively. Instead, the system account owns them. Typically, all users on the system can gain access to public volumes. The purpose of a public volume is to provide storage space for system files such as:

- Operating system files
- Utilities
- Diagnostic and test programs
- System libraries
- Help files
- Optional software

Public volumes also provide space for files that users create for public or private use.

In preparing and using private and public volumes, you must understand a number of VMS system commands. These commands allow you to prepare and gain access to a wide range of peripheral storage devices, as shown in Figure 4–1. Commands used to manage disks and tapes include:

- ALLOCATE
- DEALLOCATE
- ANALYZE/MEDIA
- ANALYZE/DISK_STRUCTURE
- INITIALIZE
- MOUNT
- DISMOUNT
- SET VOLUME
- BACKUP

Figure 4–2 reviews most of these commands. Some commands are used only for new volumes, while others are used for both new and old volumes.

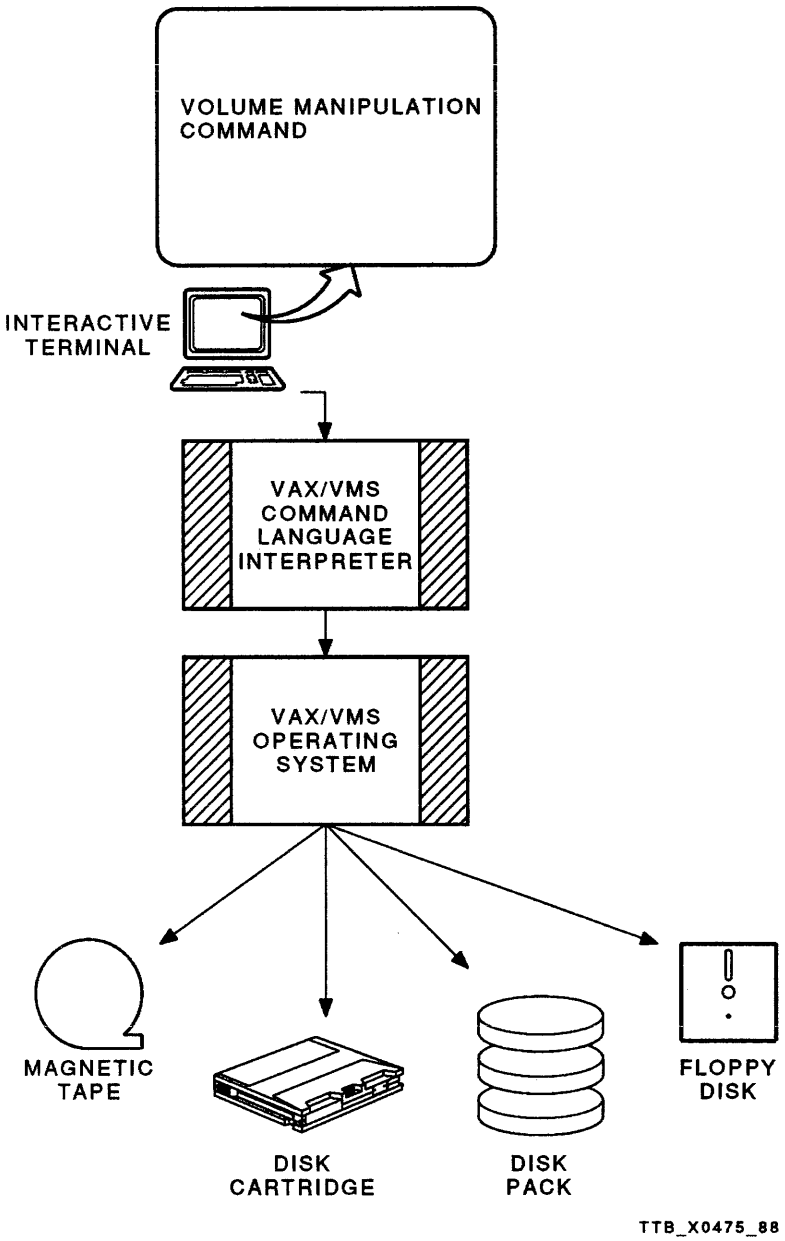


Figure 4–1 Volume Manipulation Commands

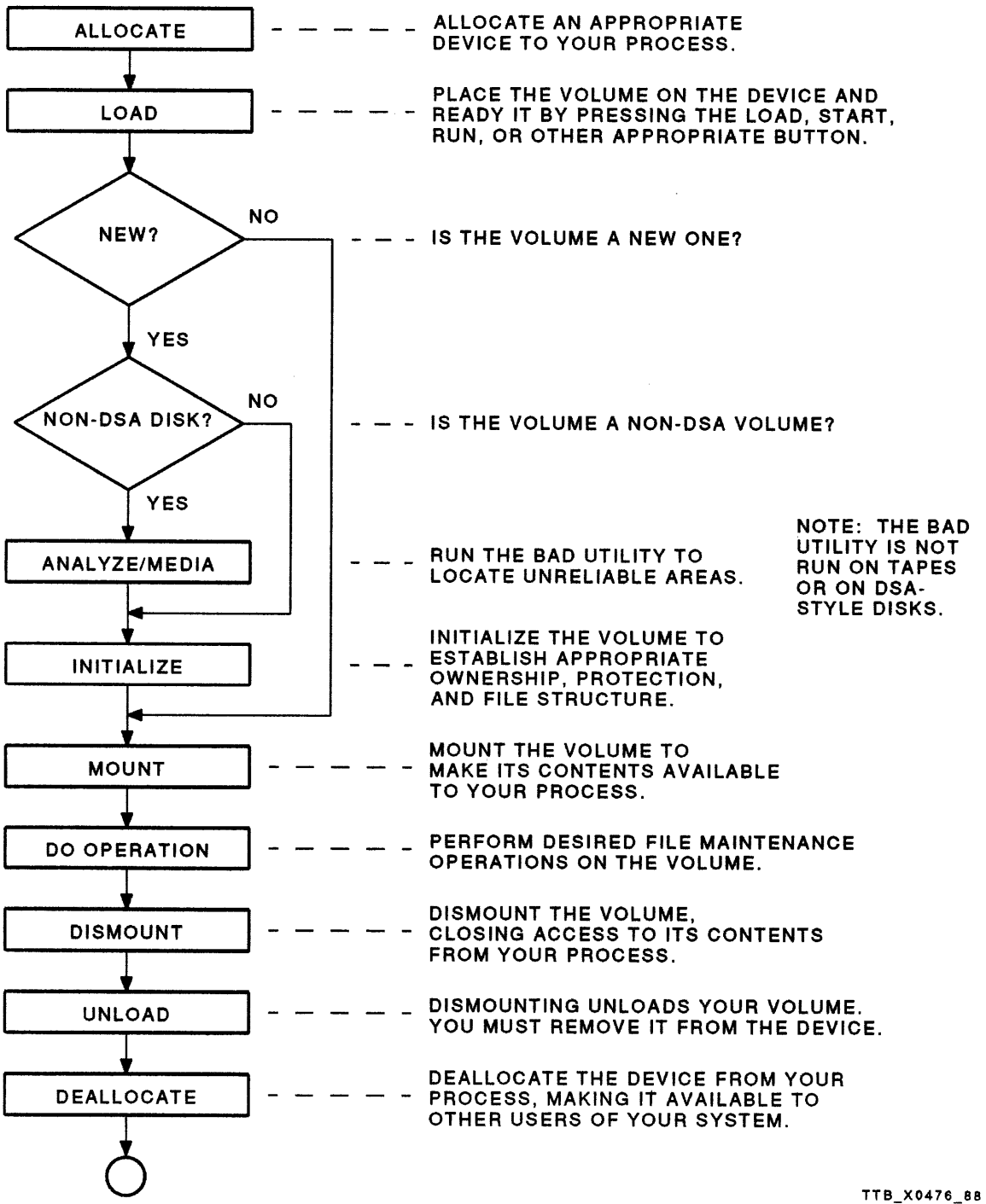


Figure 4–2 Preparing and Using a Disk or Tape Volume

4.6 Preparing Volumes for Use

4.6.1 Locating Bad Blocks

To use a disk volume, you must first format it. Digital disks are preformatted, so you do not have to perform that step.

You should, however, run the VMS Bad Block Locator Utility (BAD) on each new disk and any disk to be used for a new purpose to locate any unreliable areas on the disk. The Bad utility records the locations of the bad blocks found.

After the Bad utility locates and records the bad blocks, issue the INITIALIZE command, so the VMS system can allocate those blocks to a special file on the volume [000000]BADBLK.SYS. Once a block belongs to the bad block file, other files cannot use it.

The ANALYZE/MEDIA command invokes the Bad utility, as shown in Table 4-1.

While the system runs, the Bad utility stores the locations of unreliable blocks in the pending bad block file for the volume [000000]BADLOG.SYS. When the file that owns a bad block is deleted, that block becomes part of BADBLK.SYS.

> NOTE
> BAD should never be used on DIGITAL Storage Architecture (DSA) disks because DSA devices have a built-in utility that dynamically detects and replaces bad blocks. This utility replaces bad blocks by redirecting (revectoring) a user's access to a good, spare block. (Each DSA disk has a set of reserved spare blocks for this purpose so the user-accessible space on DSA disks is never reduced.) Also, there is no Bad utility for tapes.

Table 4–1 Bad Utility

Command	Comment
`$ MOUNT/FOREIGN DBA1:`	The volume must be mounted with the /FOREIGN qualifier.
`$ ANALYZE/MEDIA/OUTPUT=BAD.LOG DBA1:`	You can produce an output listing of the bad blocks on the volume without affecting the information stored on the volume.
`$ ANALYZE/MEDIA/EXERCISE/LOG DBA1:`	The /EXERCISE qualifier causes read/write checks to be performed on the volume. All previously stored data is destroyed. As it detects each bad block, the utility sends a message to the user because the /LOG qualifier was included.

4.6.2 Initializing and Mounting the Volume

After the disk or tape is ready to use, initialize it (using the INITIALIZE command) to define parameters such as owner, protection code, label, and cluster size. The bad block file is created on disk volumes at initialization (non-DSA disks only).

Prior to using the disk or tape volume, mount it using the MOUNT command to make it known to the system.

Once a volume is mounted, you can use it to perform operations such as:

- Copying files to or from the volume
- Creating files and directories on a disk volume
- Manipulating files on a disk volume
- Listing files on a tape or disk volume

The VMS operating system limits the type and extent of operations a user can perform on a volume according to a user's access to the volume.

4.6.3 Defining User Access to Volumes

The VMS operating system assigns a volume protection code to each disk volume similar to a file protection code. The default code is (S:RWED,O:RWED, G:RWED,W:RWED). You can override this default code by specifying a volume protection code when you initialize or mount the disk.

The VMS operating system also assigns an owner UIC to each disk volume, which is either the UIC specified with the INITIALIZE/OWNER command, or the UIC of the process that initializes the volume. The system compares the owner UIC against the UIC of each process that attempts to gain access to the volume, allowing or preventing access according to the protection code of the volume.

The VMS operating system does not assign a default protection code to tape volumes. You can assign a protection code (in the same format as the disk protection code) when you initialize or mount the tape.

The VMS operating system does assign an owner UIC to tape volumes (the UIC specified with the INITIALIZE/OWNER command or the UIC of the process that initializes the volume). However, the VMS operating system uses the owner UIC to allow or prevent access to a volume only if the tape contains a protection code. Therefore, any user can gain access to a tape that does not have a protection code.

> NOTE
> Users with the VOLPRO privilege can gain access to any disk or tape volume even if the protection code on the volume does not allow them access to it. However, these users must include the /OVERRIDE=OWNER_ID qualifier in the MOUNT command to override the VMS operating system protection checks. The MOUNT command in Example 4-4 includes this qualifier.

If you pass the disk volume protection test, the protection codes and access-control lists on specific directories and files limit your access to files on the volume. If you pass the tape volume protection test, you have access to all the files on that volume. File protection codes and access control lists are discussed in more detail in Module 8.

Figure 4-3 and Table 4-2 show the effects of protection codes on user access rights to files and volumes.

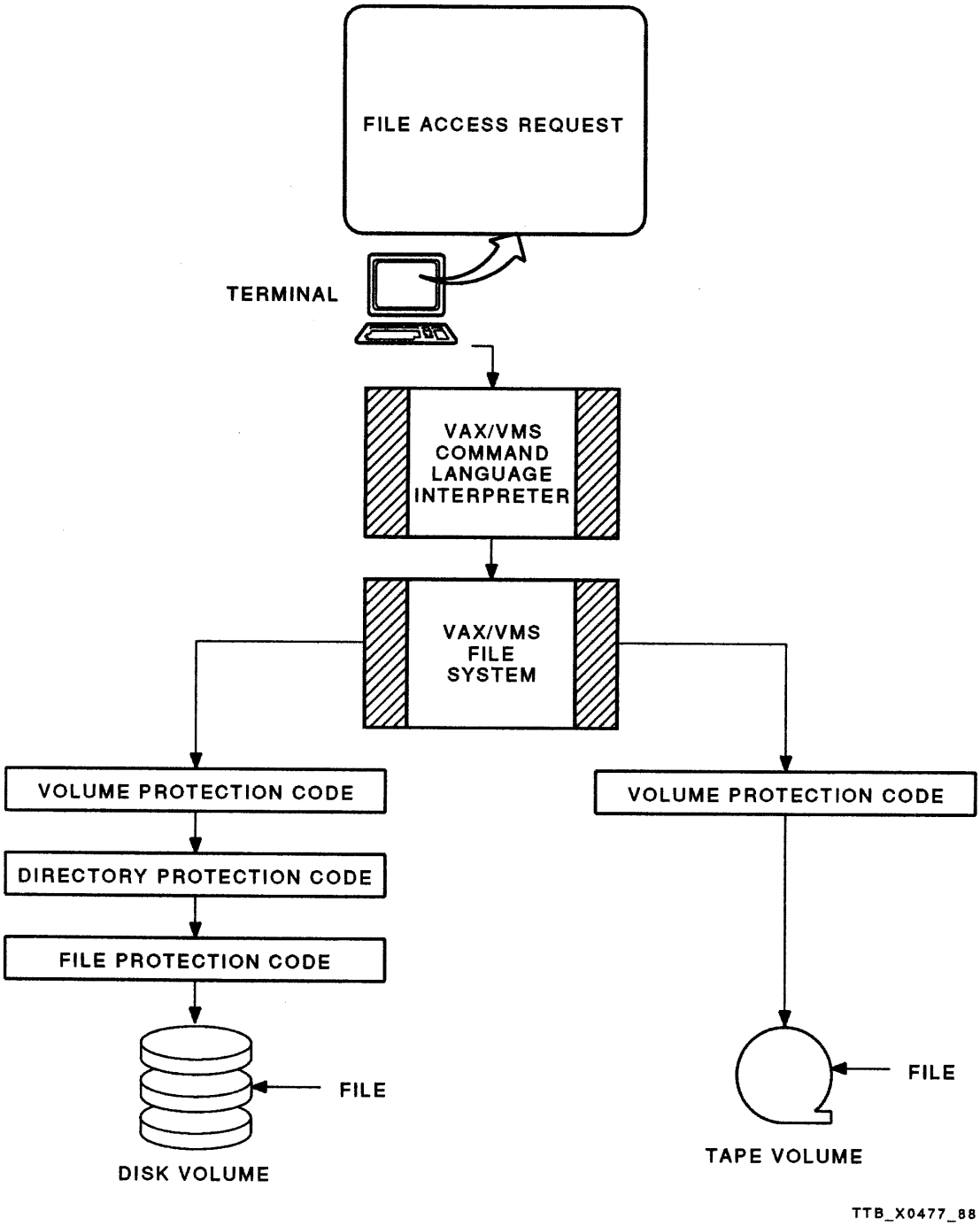


Figure 4–3 User Access to Files on Disk and Tape Volumes

Table 4–2 Effects of Access Rights to Files

(R)ead	(W)rite	(E)xecute	(D)elete	(C)ontrol†
Disk Volumes				
Reads files on the volume	Modifies files on the volume	Creates files on the volume	Deletes files from volume	Does not apply
Disk Directories				
Lists files in directory with or without using wildcards	Writes to the directory file (requires read access to create or remove files or perform other operations that change the directory)	Lists files by name only (Cannot use wildcards to list files)	Deletes the directory, if it is empty	Changes the characteristics of the directory (SET DIRECTORY)
Disk Files				
Reads contents of file (read access implies execute access)	Modifies the contents of a file	Executes executable files	Deletes or renames the file	Changes the characteristics of the file (SET FILE)
Tape Volumes				
Reads list of files on tape	Adds files on the volume (write access implies read access)	Does not apply	Does not apply	Does not apply

†Control access does not appear in the protection mask for a file. It is never granted to the GROUP or WORLD categories of users; it is always granted to SYSTEM and OWNER users.

To initialize a volume, at least one of the following must be true:

- The volume is blank (new).
- The owner UIC of the volume matches yours.
- You have the VOLPRO privilege.

To mount a volume and change the owner UIC or volume protection code, you must own it (the UIC of the volume must match the UIC of your process) or have the VOLPRO privilege.

Tables 4-3 and 4-4 illustrate the INITIALIZE command format and the qualifiers used to set the owner UIC and volume protection code. Note that you can combine the /PROTECTION and /OWNER_UIC qualifiers to define specific protection, while the /GROUP, /SYSTEM, and /NOSHARE qualifiers establish predefined protection codes.

Table 4–5 lists qualifiers to the MOUNT command that override the protection code and owner UIC established at initialization.

Table 4–3 Defining Specific Volume Protection Codes During Initialization

Qualifier	Effects	Comments
None	Owner UIC for disk: none Owner UIC for tape: UIC of current user	On disks, the owner UIC field is empty.
	Protection on disk: `(S:RWED,O:RWED,G:RWED,W:RWED)`	The default protection code allows access to all users.
	Protection on tape: none	The system does not record a protection code on tape unless you specify one.
`/PROTECTION=code`	Owner UIC: UIC of current user Protection: specified in code	The code is in the same format as file protection codes. For tapes, only Read and write access are meaningful. Also, System and Owner always have R and W access to a tape regardless of what is specified. For disks, E access allows create access. Unless you specify protection for tapes, all users have access.
`/OWNER_UIC=uic`	Owner UIC: UIC specified Protection: no affect on code	The VMS operating system compares the UIC of the process attempting access to the owner UIC and protection code of the volume.

Table 4–4 Establishing Predefined Volume Protection Codes During Initialization

Qualifier†	Effects	Comments
/NOSHARE	Owner UIC: UIC of current user Protection: (S:RWED,O:RWED,G,W)	Only system and owner have access
/GROUP/NOSHARE	Owner UIC: [g,0]‡ Protection: (S:RWED,O:RWED,G:RWED,W)	Expands access to group members as well
/GROUP	Owner UIC: [g,0]‡ Protection: (S:RWED,O:RWED,G:RWED,W:RWED)	Expands access to all users
/SYSTEM	Owner UIC: [1,1] Protection: (S:RWED,O:RWED,G:RWED,W:RWED)	Gives all users access, but only system users can create top-level directories. Public volumes are typically mounted using the /SYSTEM qualifier so the manager can govern the use of disk space.

†Cannot combine qualifiers except as shown
‡g = group number of current user

Table 4–5 Overriding Volume Protection Codes Established at Initialization

Purpose	MOUNT Command Qualifier	Effects/Required Privilege
To override volume parameters as volume is mounted:		
Override owner UIC of tape or disk	`/OWNER_UIC=uic`	Specified UIC is owner while volume is mounted (does not modify UIC written on volume) Volume ownership or VOLPRO privilege required
Override disk protection code	`/PROTECTION=code`	Specified code is protection code while volume is mounted (does not modify protection code written on volume) Volume ownership or VOLPRO privilege required

When you initialize a disk, the qualifiers to the INITIALIZE command shown in Tables 4-4 and 4-5 specify the owner UIC and protection code of the volume. When you mount a disk, you can change these established protection codes if you own the volume or have the VOLPRO privilege. Furthermore, you can include a qualifier to the MOUNT command to restrict access to a volume even if its protection code grants access to all users. For example, if the protection code of a volume grants all types of access to all users, but you mount the volume as a private volume, other users cannot gain access to it. You must mount it as a system, group, or shared volume to grant access to others (see Table 4–6).

Table 4–6 Specifying User Access to a Volume

Qualifier	Comments
Mounting a disk for private use	
none	The user must own the volume or have VOLPRO privilege. If the user has VOLPRO privilege but does not own the volume or know the label, he or she must include the /OVERRIDE=IDENTIFICATION qualifier with the MOUNT command to access the volume (see Example 4–1). If a qualifier is not included in the MOUNT command to allow access to other users, only the owner can access the volume while it is mounted regardless of the protection code.
Making a disk accessible to your group	
/GROUP	If the volume was initialized as a GROUP volume to set the protection code, but is not mounted /GROUP, group users cannot access it. Assigns logical name in group table (GRPNAM privilege required). Group members can gain access to the volume without mounting it because name is in group logical name table. Group members must still pass the volume protection code.
Making a disk accessible to all users on the system	
/SYSTEM	If the volume was initialized as a SYSTEM volume to set the protection code, but is not mounted /SYSTEM, users cannot access it. Assigns logical name in system table (SYSNAM privilege required). All users can gain access to the volume without mounting it. Users must still pass the volume protection code. Example 4-2 uses the /SYSTEM qualifier.
Making a disk available to all users on condition	
/SHARE	Other users must issue a MOUNT/SHARE command. Other users must still pass the volume protection code.

4.6.3.1 Additional References

- To learn more about the format of disk and tape volumes, consult the *Guide to VMS Files and Devices*.
- To learn more about how to run the Bad utility on disk volumes, read the *VMS Bad Block Locator Utility Manual*.
- To learn more about how the VMS operating system provides protection for disk and tape volumes, read the chapter on setting up and maintaining public volumes in the *Guide to Maintaining a VMS System*.
- To learn more about how to prepare volumes for use, read the chapter on setting up and maintaining public volumes in the *Guide to Maintaining a VMS System*.
- For more information on the ALLOCATE, INITIALIZE, and MOUNT commands, read their respective command descriptions in the *VMS DCL Dictionary*.

4.7 Obtaining and Modifying Volume Information

After a volume is mounted, you can list its characteristics by using the SHOW DEVICE command. You can modify the characteristics of a volume by using the SET VOLUME command (you must own the volume or have the VOLPRO privilege). Table 4–7 gives an example of both commands.

Table 4–7 Obtaining and Modifying Volume Information

Operation	Command	Function
Displaying the characteristics of the volume	`$ SHOW DEVICE/FULL device` `$ SHOW DEVICE/FULL MYDISK`	Displays the characteristics of the volume currently mounted on the device
Modifying the characteristics of the volume	`$ SET VOLUME/qualifier volume-name` `$ SET VOLUME/LABEL=MAY10BCK MYDISK`	Requires VOLPRO privilege or volume ownership to change the characteristics

Example 4–1 shows the output to a SHOW DEVICE command for a typical disk volume. The last section of information the command displays is volume information. Note the following characteristics:

- **Volume Label:** The disk label was specified when the disk was INITIALIZEd, VAXVMSRL052.
- **Owner UIC:** The owner UIC associated with the disk is [1,1], which indicated that the SYSTEM owns the disk. For privately mounted volumes, the UIC is the default UIC of your process.
- **Owner Process:** For private volumes, this field contains the name of the user who mounted the volume. For public volumes, this field is blank.
- **Volume Protection:** The protection code on volumes is in the same format as the protection code for files. The protection code in Example 4-1 allows any user complete access to this volume. However, a user must still pass the protection code of individual directories and files on the volume to gain access to them.
- **Mount Status:** The mount status field of the display contains the word **System**, indicating that the unit has been mounted for access by all users. If the volume had been mounted as either a private or a shared volume, the status field would contain **Process**.
- **Mount Count:** The number of processes that have mounted the volume.
- **Relative Volume Number:** The relative volume number is 0, indicating that this volume is not a member of a volume set. (Volume sets are discussed later in this module.)

```
$ SHOW DEVICE/FULL DUA0

Disk BROWNY$DUA0:, device type RA81, is online, mounted, file-oriented device,
    shareable, served to cluster via MSCP Server, error logging is enabled.

    Error count                    0    Operations completed            612665
    Owner process                  ""   Owner UIC                       [1,1]
    Owner process ID        00000000    Dev Prot    S:RWED,O:RWED,G:RWED,W:RWED
    Reference count                72   Default buffer size                512
    Total blocks               891072   Sectors per track                   51
    Total cylinders              1248   Tracks per cylinder                 14

    Volume label       "VAXVMSRL052"    Relative volume number               0
    Cluster size                   3    Transaction count                  153
    Free blocks                72309    Maximum files allowed           111384
    Extend quantity                5    Mount count                          6
    Mount status              System    Cache name      "_BROWNY$DUA0:XQPCACHE"
    Extent cache size             64    Maximum blocks in extent cache    7230
    File ID cache size            64    Blocks currently in extent cache  5313
    Quota cache size               0    Maximum buffers in FCP cache       129

  Volume status:  subject to mount verification, file high-water marking, write-
      through caching enabled.
  Volume is also mounted on CACAO, WIZTOO, CABALA, STRAD, GREEBY.
$
```

Example 4–1 SHOW DEVICE/FULL Command

4.8 Using a Public Volume

In Example 4–2, a public volume is prepared, mounted, and used. Always mount public volumes with the /SYSTEM qualifier so all users can gain access to them. The notes following the example discuss each command in greater detail. Tables 4-8 and 4-9 contain more information on each command.

```
❶ $ SHOW DEVICE DRA2

  Device                Device        Error     Volume      Free Trans Mnt
   Name                 Status        Count     Label     Blocks Count Cnt
  DRA2:                 Online            0
  $
❷ $ ALLOCATE DRA2 PUBDISK
  %DCL-I-ALLOC, _DRA2:  allocated
❸ $ MOUNT/FOREIGN PUBDISK
  %MOUNT-I-MOUNTED, USER_DISK     mounted on _DRA2:
❹ $ ANALYZE/MEDIA/EXERCISE PUBDISK
❺ $ DISMOUNT/NOUNLOAD PUBDISK
❻ $ INITIALIZE/SYSTEM PUBDISK MYVOL
❼ $ MOUNT/SYSTEM PUBDISK MYVOL
  %MOUNT-I-MOUNTED, MYVOL         mounted on _DRA2:
  $ SHOW LOGICAL/SYSTEM D*

  (LNM$SYSTEM_TABLE)

     "DBG$INPUT" = "SYS$INPUT:"
     "DBG$OUTPUT" = "SYS$OUTPUT:"
     "DDP$DIS" = "SYS$MANAGER:DDP.DIS"
     "DISK$MYVOL" = "DRA2:"
     "DISK$VAXVMSRL052" = "DUA0:"
     "DTR$LIBRARY" = "SYS$SYSROOT:[DTR]"
  $
```

Example 4–2 Initializing and Mounting a Public Disk

Notes on Example 4–2:

❶ `$ SHOW DEVICE DRA2`

The SHOW DEVICE command displays the status of devices on your system. You have requested a report on the status of the device named DRA2.

❷ `$ ALLOCATE DRA2 PUBDISK`

The ALLOCATE command causes the system to allocate the DRA2 device to your process. The system displays a message informing you that the device is allocated. After allocating the device to your process, the system assigns the device name DRA2 to the logical name PUBDISK and records the assignment in your process logical name table.

At this point, place a disk in the device and load it.

❸ `$ MOUNT/FOREIGN PUBDISK`

The MOUNT command makes the contents of your volume available to the system. The /FOREIGN qualifier indicates that the file structure on the volume is unknown. This command uses the logical name PUBDISK (defined by the preceding ALLOCATE command) to refer to the DRA2 device, but you could have used the device name DRA2 instead.

In this example, the disk pack is a used one. You can tell it is used because the MOUNT command returned a message that included its label, USER_DISK. The MOUNT command was successful because the user had the VOLPRO privilege or the UIC of the volume matched the user's UIC. Had the volume been a new one, neither command would have been required.

❹ `$ ANALYZE/MEDIA/EXERCISE PUBDISK`

This command invokes the VMS Bad Block Locator utility, which erases the disk volume and stores the location of bad blocks in a reserved area on the disk. (Do not run the Bad utility on a DSA disk.) No messages were returned to the terminal indicating that the Bad utility found no bad blocks on the volume.

> NOTE
> If you do not include the /EXERCISE qualifier, the Bad utility does not write on the disk. Instead, it produces a listing of the bad blocks already recorded on the disk.

❺ `$ DISMOUNT/NOUNLOAD PUBDISK`

Before building a file structure on the volume, you must first dismount it by issuing the DISMOUNT command. The /NOUNLOAD qualifier tells the system that you want the volume to remain accessible to your process (on-line); without it, you might have to reload the volume. Whether you reload the volume or not, you have to press the on-line switch on the drive before you can continue. You equated the logical name PUBDISK with DRA2 when you allocated the device.)

❻ `$ INITIALIZE/SYSTEM PUBDISK MYVOL`

Initializing the disk with the /SYSTEM qualifier sets the owner UIC to [1,1] and sets the protection code to (S:RWED,O:RWED,G:RWED,W:RWED). These values remain until the volume is initialized again. However, you can override them when you mount the volume by using MOUNT qualifiers (see Table 4–5). In this example, PUBDISK is a logical name assigned to the device, and MYVOL is the new volume label.

❼ `$ MOUNT/SYSTEM PUBDISK MYVOL`

Mounting the disk with the /SYSTEM qualifier deallocates the device and places the logical name in the system logical name table. (This operation requires SYSNAM privilege.) This command allows all users access to the volume without mounting it. (Note the logical name DISK$MYVOL in the system logical name table.)

Table 4–8 Creating and Accessing Private Disk and Tape Volumes

Command/Example	Function
Allocating a device	
`$ ALLOCATE device [logical-name]` `$ ALLOCATE DU DISK`	Finds the first available disk drive of type DU and allocates it to your process. The system places the logical name DISK in your process logical name table, equating it to the name of the allocated device. Other users cannot gain access to this device. Since you used the ALLOCATE command, you must use the DEALLOCATE command when you are finished to make the device available to other users.
Finding unreliable areas on an unused disk	
`$ MOUNT/FOREIGN device` `$ ANALYZE/MEDIA/EXERCISE device`	Finds unreliable areas of the disk and records their locations in a special block. The process destroys all previous data on the disk. If the disk is not your own, VOLPRO privilege is required.
`$ DISMOUNT/NOUNLOAD device`	Since the volume is mounted as a foreign volume, you must dismount it so you can initialize it as a Files-11 volume.
Creating a file structure on a tape or disk	
`$ INITIALIZE device label` `$ INITIALIZE DUA2: TESTDISK`	Builds Files-11 On-Disk Structure on the volume loaded on the DUA2: device. The volume is labeled TESTDISK. You are declared owner of the disk; all user groups are allowed all types of access (RWED). You must own the disk, or possess VOLPRO privilege to initialize it.
`$ INITIALIZE MUA0: TSTTAP`	Builds current ANSI level tape structure on the volume located on device MUA0. The volume receives the label TSTTAP. By default, you are declared the owner; all user groups are allowed all types of access (RWED).
Creating a link between the volume and your process	
`$ MOUNT device label [logical-name]`	Allocates the device to your process until you issue the DISMOUNT command.
`$ MOUNT DUA2: TESTDISK DISK`	Mounts TESTDISK on the device DUA2. The logical name DISK is assigned the equivalence name DUA2.

Table 4–9 Removing Private Disk and Tape Volumes

Command	Comments
Breaking the link between the volume and your process	
`$ DISMOUNT [/NOUNLOAD] device`	If you want the VMS operating system to keep the volume on-line when you dismount it, use the /NOUNLOAD qualifier. The DISMOUNT command does not deallocate a device that you have allocated with the ALLOCATE command.
`$ DISMOUNT DUA2:`	Dismounts and automatically unloads the volume on DUA2. Deletes the logical name DISK assigned by the MOUNT command.
Deallocating a device	
`$ DEALLOCATE device`	
`$ DEALLOCATE DUA2:`	Deallocates device DUA2. Frees the device for use by other users. Does not delete a logical name assigned by the ALLOCATE command.

4.8.1 Limiting Space Allocation on Public Volumes

The system manager can limit access to a volume with the INITIALIZE command. INITIALIZE can also govern the allocation of space on a volume (see Table 4–10).

Table 4–10 Affecting Space Allocation using INITIALIZE Command Qualifiers

Qualifier	Comment
/CLUSTER_SIZE=n	The minimum number of contiguous blocks to be allocated to each file.
/EXTENSION=n	The number of blocks to give to a file when it is extended. (ODS-1 disks only)
/MAXIMUM_FILES=n	The maximum number of files the disk volume can contain (once set, can only be changed by reinitializing the volume).
/DENSITY=n	The density (800, 1600 or 6250 bits/inch) at which a tape is to be written. If writing RX02 floppies, density is SINGLE or DOUBLE.

4.8.1.1 Additional References

- To learn more about the SHOW DEVICE and SET VOLUME commands, read their respective command descriptions in the *VMS DCL Dictionary*.
- To learn more about the qualifiers to the MOUNT and INITIALIZE commands, read the qualifier descriptions for these commands in the *VMS DCL Dictionary*.

4.9 Laboratory Exercises

You need a disk drive and a disk volume or a tape drive and a tape volume to complete these exercises.

1. Perform the following steps to initialize and mount a disk volume or a tape volume for public use:
   a. Allocate the device specified by your course administrator to your process.
   b. Load your volume on the device. If you are not sure how to load the volume, ask your course administrator.
   c. FOR DISK VOLUMES ONLY:
      Mount the disk volume using the /FOREIGN qualifier.
   d. FOR DISK VOLUMES ONLY:
      If your disk is one for which the Bad utility is valid (in other words, not an RA-series disk), invoke the Bad utility to search for bad blocks on the disk volume.
   e. FOR DISK VOLUMES ONLY:
      Dismount, but do not unload, the disk volume.
   f. Initialize the disk or tape volume so that all users have access to it.
   g. Mount the disk or tape volume so that all users on the system have access to it without having to mount it themselves.
2. Display the characteristics of the device on which you have just mounted the volume.
3. Dismount the volume and deallocate the device so that it is available for other users.

4.10 Solutions to Laboratory Exercises

Substitute the name of the device you are using for DRA1:, your own logical name for DRIVE, and your own label for MYVOL.

1. Enter the following commands to prepare a volume for public use:
   a. `$ ALLOCATE DRA1: DRIVE`
   b. No solution required.
   c. `$ MOUNT/FOREIGN DRIVE`
      (For disk volumes only)
   d. `$ ANALYZE/MEDIA/EXERCISE DRIVE`
      (For disk volumes only)
   e. `$ DISMOUNT/NOUNLOAD DRIVE`
      (For disk volumes only)
   f. `$ INITIALIZE/SYSTEM DRIVE MYVOL`
   g. `$ MOUNT/SYSTEM DRIVE MYVOL`
2. `$ SHOW DEVICE/FULL DRIVE`
3. `$ DISMOUNT DRIVE`
   `$ DEALLOCATE DRA1:`

4.11 Allocating Devices

Every system has a limited number of disk devices. The system manager must decide which devices will hold public volumes and which will be available for private volumes.

4.11.1 Private Volumes

On some systems, one drive is set aside for use with many private volumes. The manager plans a schedule to rotate the use of the drive among the users, and teaches the users proper loading and unloading techniques.

> CAUTION
> If not properly trained and monitored, users can damage this expensive equipment. If your site has properly trained operators, (through the VMS Operator course) restrict these procedures to them. (Users can enter the REQUEST command to ask operators to load and unload volumes.)

When many volumes are being used in the same drive, perform the following steps for each volume to get the best results.

1. Load the volume in the drive and execute the Bad utility (non-DSA disks only) before you initialize and put information on it. (Run the Bad utility on the volume while it is loaded in the drive where it will be used.)
2. Request Field Service to align the drive periodically and check it for malfunctions. This can prevent head crashes and other problems.

> NOTE
> The term **head crash** refers to a specific type of disk drive failure. Disk drives contain disk heads that read and write data on the disk (see the discussion of disk drives in Module 1). Because these heads will damage any disk volume they touch, a cushion of air separates them from the volume's surface. A variety of problems (dirt, power failure, disk drive malfunction, drive and volume misalignment) can cause one or more heads to touch or drag along a surface of a volume. This is referred to as a "head crash." The damage is usually quite costly to repair.

3. Allow only properly trained personnel to load and unload volumes on the drive.

4. Back up each volume before removing it from the drive to prevent the following situation:

   A user loads a volume in a misaligned drive and writes information to it. No one backs up the volume before the user removes it. While the volume is not in the drive, Field Service personnel align the drive. When the user loads the volume later, he or she may not be able to read the information stored on it.

   Since this situation can easily occur on systems where many users load volumes in the same device, remind users to back up their disk volumes to tape before removing them. If you do not allow users access to the drives, establish procedures for your operators so they will always back up private volumes for users before removing them.

On other systems, enough drives are available to keep private volumes mounted indefinitely. However, you should continue to:

1. Run the Bad utility (non-DSA disks only) on volumes while they are loaded in the drive where they will be used.

2. Periodically align the drives.

3. Allow only trained personnel to handle the media.

4. Remind users to save changes to their data and programs.

4.11.2 Public Volumes

The system manager must decide where system software and user software should reside on public volumes. System software should be on a medium- or high-speed disk on a separate controller from the data disks to achieve the best performance (although it can be stored on the same disk as user software). The manager should strive to keep the system disk less than three-quarters full to achieve maximum performance.

Managers should be careful to protect sensitive files and directories on public volumes from access by unauthorized users. For example, many system files must allow read and execute access to WORLD users, but they should not allow write access to them. Also, most users do not need read access to system directories (allowing them to list the contents of those directories) or to particularly sensitive files (such as SYSUAF.LIS).

Judicious use of volume protection codes, file protection codes, and access control lists should give the manager the necessary level of security on public volumes.

The manager must decide what optional software is needed, and whether it should reside on the main system disk or on another public disk. When obtaining a new software product, the manager must determine how much space it requires, and whether the system needs a new drive to accommodate the product.

User files also need room. Managers must allocate space on public disks for users, or assign each user or each group its own private disk.

A group manager can create directories for group users on the group disk, saving the system manager some work.

If all users share space on one public disk, the system manager should create their master file directories and consider using the DISKQUOTA subcommands in the SYSMAN utility to limit their use of the space on the volume. Module 2 discusses creating MFDs and using the SYSMAN utility in more detail.

At some sites, all users work on the same database. The manager should restore this database to a specified volume or volume set and be sure all users can gain access to it (by setting the protection codes properly).

4.11.3 Creating Volume Sets

If the database or user directories become too large to fit on any one volume, you can create a volume set.

A volume set consists of two or more disk volumes (not including the system volume) that are bound together with the MOUNT/BIND command. The system treats a volume set as one large volume. It stores files on any volume in the set that has free space. The system attempts to use space evenly over all volumes in a set.

You can bind new volumes together to create a volume set, or you can bind new volumes to existing volumes. Use the following procedure to create a disk volume set.

1. Allocate the necessary devices, and physically load the volumes on the devices.
2. Initialize each new volume in the set.
3. Use the MOUNT/BIND command to create the volume set. For example:

```
$ MOUNT/BIND=MASTER_SET DB1:,DB2:  PAYVOL1,PAYVOL2
```

   The name of this volume set is MASTER_SET. PAYVOL1 (DB1:) and it becomes the root volume because it is listed first. Its master file directory (MFD) contains the directory structure for the entire volume set. The MOUNT program binds the second volume, PAYVOL2 (DB2:) to the first volume to create the volume set.

Example 4–3 illustrates how to create a public volume set (USER SET) starting with an existing volume (USER1) as the root volume. The notes on Example 4-3 contain additional information about these commands.

```
❶ $ ALLOCATE DRA2 DEV1
  %DCL-I-ALLOC, _DRA2: allocated
  $ ALLOCATE DRA3 DEV2
  %DCL-I-ALLOC, _DRA3: allocated
❷ $ INITIALIZE/SYSTEM DEV2 USER2
❸ $ MOUNT/SYSTEM/BIND=USER_SET DEV1,DEV2 USER1,USER2
  %MOUNT-I-MOUNTED, USER1        mounted on _DRA2:
  %MOUNT-I-MOUNTED, USER2        mounted on _DRA3:
❹ $ SHOW LOGICAL/SYSTEM D*

  (LNM$SYSTEM_TABLE)

     "DBG$INPUT" = "SYS$INPUT:"
     "DBG$OUTPUT" = "SYS$OUTPUT:"
     "DDP$DIS" = "SYS$MANAGER:DDP.DIS"
     "DISK$USER1" = "DRA2:"
     "DISK$USER2" = "DRA3:"
     "DISK$USER_SET" = "DRA2:"
     "DISK$VAXVMSRL052" = "DUA0:"
     "DTR$LIBRARY" = "SYS$SYSROOT:[DTR]"
  $
❺ $ COPY WORK1:[BROWN]EXAMP5.COM
  _To: DISK$USER_SET:[SMITH]EXAMP5.COM
  $ DIRECTORY DISK$USER_SET:[SMITH]

  Directory DISK$USER_SET:[SMITH]

  EXAMP5.COM;1

  Total of 1 file.
❻ $ DISMOUNT DEV1
  $ SHOW LOGICAL/SYSTEM D*

  (LNM$SYSTEM_TABLE)

     "DBG$INPUT" = "SYS$INPUT:"
     "DBG$OUTPUT" = "SYS$OUTPUT:"
❼    "DDP$DIS" = "SYS$MANAGER:DDP.DIS"
     "DISK$VAXVMSRL052" = "DUA0:"
     "DTR$LIBRARY" = "SYS$SYSROOT:[DTR]"
  $
```

Example 4–3 Creating a Volume Set from an Existing Volume

Notes on Example 4–3:

❶
```
$ ALLOCATE DRA2: DEV1
$ ALLOCATE DRA3: DEV2
```

These commands allocate two disk devices for the volumes USER1 and USER2, and give them the logical names DEV1 and DEV2 respectively. After allocating the devices, the user loads the volumes (USER1 and USER2) into their respective drives. (USER1 is an existing volume; USER2 is a new volume.)

❷ `$ INITIALIZE/SYSTEM DEV2 USER2`

USER2, since it is a new volume, is initialized to delete old files and create a Files-11 structure. The /SYSTEM qualifier sets the owner UIC to [1,1] and the protection code to (S:RWED,O:RWED,G:RWED,W:RWED).

❸ `$ MOUNT/SYSTEM/BIND=USER_SET DEV1,DEV2 USER1,USER2`

The MOUNT command string binds the volumes into the disk volume set, USER_SET. The /SYSTEM qualifier makes the set public. You must have the SYSNAM privilege to use it. The root volume (USER1) contains the directory structure for the entire volume set.

❹ `$ SHOW LOGICAL/SYSTEM D*`

The user did not include a logical name for the volume set in the MOUNT command. Therefore, the system creates the logical names DISK$USER1, DISK$USER2, and DISK$USER_SET by default.

If you mount USER_SET at a later time, you could include a logical name for the set in the MOUNT command. In that case, the system would not create the default logical name in the form DISK$volume_set_name. The following command includes the logical name USERS, which can be used as the name of the volume set in subsequent commands.

```
$ MOUNT/SYSTEM  DEV1,DEV2  USER1,USER2  USERS
```

❺
```
$ COPY WORK1:[BROWN]EXAMP5.COM
_To: DISK$USER_SET:[SMITH]EXAMP5.COM
```

The user copies a file from a work disk to a directory on the volume set. The [SMITH] directory already existed on the USER1 volume. It is now a directory on the volume set. The system stores the file on the volume in the set that has the most unused space.

❻ `$ DISMOUNT DEV1`

To dismount an entire volume set, use the DISMOUNT command and specify any one of the devices containing a member of the volume set. To dismount a single volume of a volume set, use the /UNIT qualifier. Since the volume set was mounted using the /SYSTEM qualifier, the devices have already been deallocated. Therefore, they are now available for other users.

❼ When you dismount a volume set, the system deletes the logical names it created during the mount procedure.

For more information on creating and using disk volume sets, read Module 2 in the *Guide to Maintaining a VMS System*.

4.12 Laboratory Exercises

You need two disk drives and two disk volumes to complete the following exercises. If you do not have the necessary materials to complete the lab, write out the answers on paper and compare them to the solutions.

1. Perform the following steps to create a public volume set using two disk volumes.
   a. Allocate the devices you will be using.
   b. Load the volumes onto their respective drives. If you are not sure how to do this, ask your course administrator.
   c. Initialize the volumes.
   d. Mount the volumes using the /SYSTEM and /BIND qualifiers.
2. Create a user directory on the volume set.
3. Copy a file from your default directory to your directory on the volume set.
4. Display the contents of your directory on the volume set.
5. Dismount the volume set.

4.13 Solutions to Laboratory Exercises

Substitute the names of your devices and your own logical names and labels for the names and labels shown in the solutions.

1. Enter the following commands to create a volume set for public use:

   a.
   ```
   $ ALLOCATE DRA1: DEV1
   $ ALLOCATE DRA2: DEV2
   ```

   b. No solution required.

   c.
   ```
   $ INITIALIZE/SYSTEM DEV1 USER1
   $ INITIALIZE/SYSTEM DEV2 USER2
   ```

   d.
   ```
   $ MOUNT/SYSTEM/BIND=USER_SET DEV1,DEV2 USER1,USER2
   ```

2.
   ```
   $ CREATE/DIRECTORY USER_DISK:[BROWN]
   ```

3.
   ```
   $ SET DEFAULT SYS$LOGIN
   $ COPY TESTFILE.LIS
   _To: USER_DISK:[BROWN]*.*
   ```

4.
   ```
   $ DIRECTORY USER_DISK:[BROWN]
   ```

5.
   ```
   $ DISMOUNT DEV1
   ```

4.14 Maintaining Private and Public Volumes

Performing regular maintenance on volumes can improve their performance.

System managers should maintain all public volumes by periodically backing them up (making copies of changes) and checking for lost files.

Managers are usually not responsible for private volumes. Users should maintain their own backups of private volumes. However, at some sites, operators maintain both public and private volumes. In this case, the operator should have the VOLPRO privilege to mount disks with unknown labels.

The MOUNT/OVERRIDE command in Example 4-4 shows how to mount a volume with an unknown label.

```
❶ $ MOUNT/OVERRIDE=IDENTIFICATION DRA2 UNKNOWN MYDISK
  %MOUNT-I-MOUNTED, PROGRAM_DISK mounted on _DRA2:
❷ $ SHOW DEVICE/FULL MYDISK

Disk DRA2:, device type RM03, is online, allocated, deallocate on dismount,
        mounted, error logging enabled.

    Error count                    6    Operations completed            2643
    Owner process          "BIERLY"    Owner UIC                    [11,340]
    Owner process ID       00000094    Dev Prot S:RWED,O:RWED,G:RWED,W:RWED
    Reference count                2    Default buffer size              512

    Volume label    "PROGRAM_DISK"     Relative volume no.                0
    Cluster size                   3    Transaction count                  1
    Free blocks               131589    Maximum files allowed          16460
    Extend quantity                5    Mount count                        1
    Mount status             Process    Cache name           "_DRA0:XQPCACHE"
    File ID cache size            64    Extent cache size                 64
    Quota cache size               0
    Write-thru caching enabled

Volume is subject to mount verification, file high-water marking.

$ DISMOUNT MYDISK
$
```

Example 4–4 Mounting a Disk with an Unknown Label

Notes on Example 4–4:

❶ `$ MOUNT/OVERRIDE=IDENTIFICATION DRA2: UNKNOWN MYDISK`

The MOUNT/OVERRIDE command successfully mounts a disk when you do not specify a label or when you specify the wrong label (as in this example). The MOUNT command assigns the logical name MYDISK to the device on which the system loads the volume (in this case, DRA2:). The message returned to your terminal reports the label value, which is recorded on the volume (in this case, PROGRAM_DISK).

❷ `$ SHOW DEVICE/FULL MYDISK`

The listing generated by the SHOW DEVICE/FULL command confirms that the actual label of the volume (PROGRAM_DISK) is identical to the one reported in the MOUNT message at your terminal.

Use the Backup utility to back up files, directory structures, or entire volumes. You could use the COPY command to back up files, directories, or volumes, but the Backup utility is faster.

There are two versions of the Backup utility: on-line BACKUP and standalone BACKUP. Any user can enter on-line BACKUP commands from any terminal where he or she is logged in to back up files, directory structures, or private volumes. (To back up an entire volume, the user must mount it as a private volume first so other users cannot modify information on it during the backup procedure.) Only users who are able to shut down the system and start it again can use standalone BACKUP. Typically, the system manager or operator uses standalone BACKUP to back up the public disk that contains the system files (the system disk). The following section discusses on-line BACKUP commands in more detail. Module 7 discusses standalone BACKUP.

Use the Verify utility to check the validity of the file structure on a volume, and to locate and recover lost files and blocks allocated to improperly closed files. The Verify utility is discussed later in this module.

4.14.1 On-Line BACKUP

On-line BACKUP commands include those that allow you to save:

- Individual files (and their locations in a directory structure)
- A directory structure and the files in it
- All directory structures and files on a volume

The Backup utility stores these files and directories in a special file called a **save set**. The format of a save-set file is known only by the Backup utility. 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 [BROWN] and save FILE.DAT itself.

You can create a save-set file on a disk or tape volume. A save set is a normal VMS system file, with a standard VMS file name. You can copy it, rename it, or delete it using the same DCL commands you use to manage other VMS operating system files. However, you can only list its contents or restore its contents to another volume with the Backup utility.

The format of a BACKUP command is:

```
$ BACKUP input-specifier output-specifier
```

Table 4–11 explains the parts of this command and defines common terms used in the following discussions about the Backup utility.

Table 4–11 BACKUP Terms

Term	Definition	Comments
Save set	A file in BACKUP format, created by the Backup utility	The Backup utility can create save sets on tape or disk volumes.†
Save-set-name	Any legal VMS operating system file name, file type ‡ and version number	`SAVE.BCK;2`
Save-set-specifier	A device name and a save-set-name. The format is device:save-set-name.	`MUA0:SAVE.BCK;2`

Input-specifier	The input file specification (what you are saving) or save-set-specifier (what you are restoring from) in the BACKUP command	`$ BACKUP -`
		`_$ [BROWN]FILE.DAT -`
		`_$ output-specifier`
		
		`$ BACKUP -`
		`_$ MUA0:SAVE.BCK;2 -`
		`_$ output-specifier`

Output-specifier	The output save-set-specifier (where you are saving to) or file specification (where you are restoring to) in the BACKUP command	`$ BACKUP -`
		`_$ input-specifier -`
		`_$ MUA0:SAVE.BCK`
		
		`$ BACKUP -`
		`_$ input-specifier -`
		`_$ DBA1:[BROWN]`

† If you are creating or using a save set on a disk volume, you must follow the name of the save set with the /SAVE_SET qualifier.

‡ There is no default file type. However, the .BCK file type is used by convention.

4.14.1.1 BACKUP Qualifiers

BACKUP has five types of qualifiers:

- Command qualifiers
- Input file-selection qualifiers
- Input save-set qualifiers
- Output file qualifiers
- Output save-set qualifiers

Definitions of these qualifier types are explained in Table 4-12. Some qualifiers exist in more than one category, as shown in Tables 4-13 through 4-18.

Table 4–12 BACKUP Qualifier Types

Qualifier Type	Function
Command qualifier	Modifies the default action of a BACKUP command. You can place this type of qualifier anywhere on the command line. The qualifier acts upon every file in the input or output specifier (see Table 4–13).
Input file-selection qualifier	Selects files from the input specifier. Place immediately after the input specifier (see Table 4–15).
Input save-set qualifier	Affects the way BACKUP handles an input save set during a restore operation. Place immediately after the input specifier (see Table 4–16).
Output file qualifier	Changes the way output files are restored. Place them immediately after the output specifier (see Table 4–17).
Output save-set qualifier	Affects the way BACKUP processes an output save set during a save operation. Place immediately after the output specifier (see Table 4–18).

Table 4–13 BACKUP Command Qualifiers

Qualifier	Function
/[NO]ASSIST	Allows operator or user intervention if a request to mount a magnetic tape fails during a BACKUP process.
/BRIEF	Produces an abbreviated listing of files in the save set when used with /LIST.
/BUFFER_COUNT=n	Specifies the number of I/O buffers to be used in the BACKUP operation.
/COMPARE	Compares the save set, device, file, or files specified by the input specifier with the save set, device, file, or files specified by the output specifier and displays an error message if it finds a difference. Can be used with the /IMAGE and /PHYSICAL qualifiers.
/DELETE	Specifies that a BACKUP save or copy operation is to delete the selected input files from the input volume after **all files** have been processed.
/FAST	Processes the input specifier using a fast file scan to reduce processing time. The input specifier must be a Files-11 disk.
/FULL	Lists the file information produced by the /LIST command qualifier in the format provided by the DCL command DIRECTORY/FULL.
/IGNORE=option	Specifies that a BACKUP save or copy operation is to override certain restrictions. See Table 4–14 for list of options.
/IMAGE	Directs BACKUP to process an entire volume or volume set (all files).
/INCREMENTAL	Allows you to restore an incremental save set.
/[NO]INITIALIZE	Initializes an output disk volume, destroying all previous contents.

Table 4–13 (Cont.) BACKUP Command Qualifiers

Qualifier	Function
/INTERCHANGE	Directs BACKUP to process files in a manner suitable for data interchange with utilities and systems that are incompatible with the standard BACKUP format.
/JOURNAL[=file-spec]	Specifies that a BACKUP save operation is to create a BACKUP journal file, or append information to an existing BACKUP journal file.
/LIST[=file-spec]	Lists information about a BACKUP save set and its contents.
/[NO]LOG	Directs BACKUP to display the file spec of each file processed.
/PHYSICAL	Directs BACKUP to ignore any file structure on the input volume and to process the volume in terms of logical blocks.
/RECORD	Directs BACKUP to write the current date and time in the BACKUP date field of each file header record once a file is successfully saved or copied.
/[NO]TRUNCATE	Controls whether a copy or restore operation truncates a sequential output file at the end-of-file (EOF) when restoring it.
/VERIFY	Specifies that the contents of the output specifier be compared with the contents of the input specifier after a save, restore, or copy operation is completed.
/VOLUME=n	Indicates that a specific disk volume in a disk volume set is to be processed. (Only valid with the /IMAGE qualifier.)

Table 4–14 BACKUP /IGNORE Qualifier Options

Option	Function
INTERLOCK	Processes files that otherwise could not be processed because of file access conflicts. Used to save or copy files currently open for writing, such as when doing a backup save operation on a disk currently being used. Note that no synchronization is made with the process writing the file, so the file data that is copied to the output specifier might be inconsistent with the input file, depending on the circumstances. When a file open for writing is processed, BACKUP issues the message: `%BACKUP-W-ACCONFLICT, "file" is open for write by another user` The INTERLOCK option is especially useful if you have files that are open so often that they might not otherwise be saved. This option requires the SYSPRV privilege, a system UIC, or ownership of the volume.
LABEL_PROCESSING	Saves or copies the contents of files to the specified magnetic tape volume regardless of the information contained in the volume header record. BACKUP does not verify the volume label or expiration date before writing information to the tape volume.
NOBACKUP	Causes BACKUP to save files marked with the NOBACKUP flag by the DCL command SET FILE/NOBACKUP. If you do not specify this option, BACKUP saves only the file header record of files marked with the NOBACKUP flag.

Table 4–15 BACKUP Input File-Selection Qualifiers

Qualifier	Function
/BACKUP	Selects files according to the BACKUP date written in the file header record by the BACKUP/RECORD command.
/BEFORE=date	Selects files dated earlier than the specified date and time.
/BY_OWNER=uic	Selects files for processing according to the specified UIC.
/CONFIRM	Prompts for confirmation to process the input file.
/CREATED†	Selects files according to the value of the creation date field in each file header record.
/EXCLUDE=(file-spec,...)	Does not process files that otherwise meet the selection criteria.
/EXPIRED†	Selects files according to the value of the expiration date field in each file header record.
/MODIFIED†	Selects files according to the value of the modified date field in each file header record.
/SINCE=date	Selects files dated equal to, or later than, the specified date and time.

†Must also specify either the /BEFORE or /SINCE qualifiers

Table 4–16 BACKUP Input Save Set Qualifiers

Qualifier	Function
/[NO]CRC	Specifies that the software Cyclic Redundancy Check (CRC) is to be performed.
/[NO]REWIND	Rewinds the input tape reel to the beginning-of-tape (BOT) marker before reading the input volume.
/SAVE_SET	Directs BACKUP to treat the input file as a BACKUP save set and not as an input file to be saved.
/SELECT=(file-spec,...)	Selects the specified files for processing.

Table 4–17 BACKUP Output File Qualifiers

Qualifier	Function	
/BY_OWNER[=option]	Redefines the owner UIC for restored files. Options include:	
	default	Sets the owner UIC to the user's current default UIC.
	ORIGINAL	Retains the owner UIC of the file being restored.
	PARENT	Sets the owner UIC to the owner UIC of the directory to which the file is being restored or copied.
	[uic]	Sets the owner UIC to the UIC specified.
/NEW_VERSION†	Creates a new version of a file if a file with an identical specification already exists at the output location.	
/OVERLAY†	Writes the input file over a file with an identical specification at the output location.	
/REPLACE†	Replaces a file on the output specifier with an identically named file from the input specifier.	

†If you do not specify /NEW_VERSION, /OVERLAY, or /REPLACE, and the version number of the file being restored is identical to that of an existing file, BACKUP reports an error and does not restore the file.

Table 4–18 BACKUP Output Save Set Qualifiers

Qualifier	Function
/BLOCK_SIZE=n	Specifies the output block size in bytes for data records in a BACKUP save set.
/BY_OWNER=uic	Specifies the owner UIC of the save set.
/COMMENT=string	Places a comment of up to 1024 characters in an output save set.
/[NO]CRC	Specifies that the software Cyclic Redundancy Check (CRC) is to be computed and stored in the data blocks of the output save set.
/DENSITY=n	Specifies the recording density of the output magnetic tape. Note that /REWIND is required with this qualifier.
/GROUP_SIZE=n	Defines the number of blocks BACKUP places in each redundancy group.
/LABEL=(string[,...])	Specifies the one- to six-character volume labels for magnetic tapes to which the save set is written.
/OWNER_UIC=uic	Specifies the owner UIC of the save set.
/PROTECTION=code	Specifies the protection to be applied to the save set.
/[NO]REWIND	Rewinds the output tape to the beginning-of-tape (BOT) marker and initializes the output tape. Specifying /NOREWIND causes the tape to wind forward to the logical end-of-tape (EOT) and to begin writing the save set there.
/SAVE_SET	Directs BACKUP to treat the output file as a BACKUP save set (on disk).
/TAPE_EXPIRATION[=date]	Writes the date on which the tape will expire to the volume header record. /REWIND must also be specified with this qualifier.

Table 4–19 Saving Files and Directories with On-Line BACKUP

Example	Comments
`$ BACKUP [BROWN]FILE.DAT -` `_$ MUA0:SAVE.BCK`	You must include the name of a save set as either the input specifier or the output specifier. To create a save set on a tape volume, mount the volume with the MOUNT/FOREIGN command first, so the Backup utility can format the tape.

`$ BACKUP [BROWN]*.*;*, -`
`_$ [SMITH]PRGM.FOR -`
`_$ MUA0:SAVEALL.BCK`
	You can save several files by including wildcards and/or listing the names (separated by commas). However, you can only specify one save set in the output specifier.

`$ BACKUP [BROWN]FILE.DAT -` `_$ DRA1:[ARCHIVE]BROWN.BCK/SAVE_SET`	Include the /SAVE_SET qualifier after the save-set name if the save set is on a disk volume. You can create save sets on any disk volume where you can create other files. You do not have to mount the volume as a foreign volume first.

To list the contents of a save set, use the /LIST qualifier as follows:

```
$ BACKUP/LIST save-set-specifier
```

In Example 4–5, an on-line BACKUP command with the /LIST qualifier lists the contents of a save set. Request a directory of the volume to find out the name of the save set(s) on the volume. Notice that you must dismount it and remount it as a foreign volume before using the Backup utility.

```
$ ALLOCATE MUA0
%DCL-I-ALLOC, _MUA0:  allocated
$ MOUNT/OVERRIDE=ID MUA0
%MOUNT-I-MOUNTED, SAVE mounted on _MUA0:
$ DIRECTORY MUA0:

Directory MUA0:[]

SAVE.BCK;1

Total of 1 file.
$DISMOUNT/NOUNLOAD MUA0
$ MOUNT/FOREIGN MUA0
%MOUNT-I-MOUNTED, SAVE mounted on _MUA0:
$ BACKUP/LIST MUA0:SAVE.BCK
Listing of save set

Save set:          SAVE.BCK
Written by:        BROWN
UIC:               [011,340]
Date:              19-APR-1989 11:18:59.32
Command:           BACKUP WORK1:[BROWN]*.*;* MUA0:SAVE.BCK
Operating System:  VAX/VMS version 5.2
BACKUP version:    V5.2
CPU ID register:   0138700D
Node Name:         _SUPER::
Written on:        _MUA0:
Block size:        8192
Group size:        10
Buffer count:      3

[BROWN]A.LOG;6                  3 15-NOV-1988  11:43
[BROWN]FILE.DAT;1               1 17-OCT-1988  13:02
[BROWN]FILE.DAT;2               1 17-OCT-1988  14:00
[BROWN]JUNK.DAT;9               1 28-JUL-1988  09:00
[BROWN]MEMO.DAT;2               2  4-FEB-1989  10:49

Total of 5 files, 8 blocks
End of save set

$ DISMOUNT MUA0
$ DEALLOCATE MUA0:
```

Example 4–5 Listing the Contents of a Save Set

To restore the contents of a save set, enter the save-set specifier as the input specifier, and the name of a directory as the output specifier. Table 4–20 lists the only acceptable directory name formats. You must enter the directory name in one of these formats. Each format produces a different result on the output volume.

You can restore specific files using the /SELECT qualifier. In Example 4–6, all the files in the [BROWN] directory are copied to a save set on the tape volume mounted on the MUA0 drive. One file from that save set is then restored to the [SMITH] directory on the volume mounted in the DRA2 drive.

Table 4–20 Restoring Save Sets with On-Line BACKUP

Output Specifier Format†	Function
`[*...]`	Restores the directory structure and files in the save set to the output volume in their original form.

`$ BACKUP -`
`_$ MUA0:SAVE.BCK -`
`_$ DRA1:[*...]`
	Restores the files in the save set to DRA1:[BROWN] and DRA1:[BROWN.FILES], respectively. If these directories do not exist on the volume in DRA1, the Backup utility creates them.

`[...]`	Restores the files in the save set to the current default directory and subdirectories of the current default directory.

`$ SHOW DEFAULT`
`DRA1:[SMITH]`
`$ BACKUP -`
`_$ MUA0:SAVE.BCK -`
`_$ [...]`
	Restores the files from the [BROWN] directory to DRA1:[SMITH]. Restores the files from [BROWN.FILES] to [SMITH.FILES]. The utility creates subdirectories as needed.

`[directory...]`	Restores the files in the save set to the named directory and subdirectories of the named directory.

`$ BACKUP -`
`_$ MUA0:SAVE.BCK -`
`_$ DRA1:[JONES...]`
	Restores the files from the [BROWN] directory to DRA1:[JONES]. Restores the files from [BROWN.FILES] directory to [JONES.FILES]. The utility creates the [JONES] and [JONES.FILES] directories if they do not exist.

`[directory]`	Restores the files in the save set to the named directory. If the save set contains files within subdirectories, the subdirectories are not created; instead, all files in all subdirectories are restored to the single named directory.

`$ BACKUP -`
`_$ MUA0:SAVE.BCK -`
`_$ DRA1:[JONES]`
	Restores the files from the save set to the [JONES] directory. All files in the directory and its subdirectories in the save set are restored as well. The utility does not create any subdirectories.

`[ ]` or no directory	Restores the files in the save set to the current default directory.

†You are restricted to these formats when you restore save sets. For all examples in this table, the save set contains the directory [BROWN], the subdirectory [BROWN.FILES], and several files in each.

Table 4–20 (Cont.) Restoring Save Sets with On-Line BACKUP

Output Specifier Format†	Function

`$ SHOW DEFAULT`
`DRA1:[SMITH]`
`$ BACKUP -`
`_$ MUA0:SAVE.BCK -`
`_$ []`
	Restores all files in the save set to the DRA1:[SMITH] directory. The utility does not create subdirectories.

`$ BACKUP -`
`_$ MUA0:SAVE.BCK -`
`_$ *.*`	

†You are restricted to these formats when you restore save sets. For all examples in this table, the save set contains the directory [BROWN], the subdirectory [BROWN.FILES], and several files in each.

```
$ ALLOCATE MUA0
%DCL-I-ALLOC, _MUA0:  allocated
❶ $ INITIALIZE MUA0 BROWN
$ MOUNT/FOREIGN MUA0
%MOUNT-I-MOUNTED, BROWN mounted on _MUA0:
$ SET DEFAULT DRA1:[BROWN]
$ BACKUP *.*;* MUA0:BROWN.BCK
❷ $ BACKUP/REWIND/LIST MUA0:BROWN.BCK
Listing of save set

Save set:           BROWN.BCK
Written by:         BIERLY
UIC:                [011,340]
Date:               19-APR-1989 11:08:51.55
Command:            BACKUP *.*;* MUA0:BROWN.BCK
Operating system:   VAX/VMS version 5.2
BACKUP version:     V5.2
CPU ID register:    0138700D
Node name:          _SUPER::
Written on:         _MUA0:
Block size:         8192
Group size:         10
Buffer count:       3

[BROWN]EXAMP5.COM;2                          1 17-NOV-1988  14:19
[BROWN]EXAMP5.COM;1                          1 17-NOV-1988  14:18
[BROWN]EXAMP5.DAT;4                          4 17-NOV-1988  14:24
[BROWN]EXAMP5.DAT;3                          4 17-NOV-1988  14:23
[BROWN]EXAMP5.DAT;2                          4 17-NOV-1988  14:19
[BROWN]EXAMP5.DAT;1                          4 17-NOV-1988  14:19
[BROWN]MAIL.MAI;1                           29 28-MAY-1989  17:06

Total of 7 files, 47 blocks
End of save set

❸ $ DIRECTORY DRA2:[SMITH]
%DIRECT-W-NOFILES, no files found
$ BACKUP/REWIND MUA0:BROWN.BCK/SELECT=EXAMP5.COM DRA2:[SMITH]
$ DIR DRA2:[SMITH]

Directory DRA2:[SMITH]

EXAMP5.COM;2        EXAMP5.COM;1

Total of 2 files.
$
```

Example 4–6 Restoring Specific Files from a Save Set

Notes on Example 4–6:

❶ `$ INITIALIZE MUA0: BROWN`

You do not need to initialize a new tape. If the tape already contains save sets, the Backup utility adds new save sets after the old ones by default. If you initialize the tape, the utility writes your save sets at the beginning of the tape.

❷ `$ BACKUP/REWIND/LIST MUA0:BROWN.BCK`

If you do not include the /REWIND qualifier, the Backup utility cannot find the save set because it only searches in a forward direction. The /REWIND qualifier causes the utility to rewind the tape to the beginning before starting the search.

❸ `$ DIRECTORY DRA2:[SMITH]`

The Backup utility restored both versions of the file EXAMP5.COM because the previous command did not specify any version number.

You can also use on-line BACKUP to save the contents of an entire public or private volume. Although you could list all the directories on a volume (using wildcards) to perform this task, using the /IMAGE qualifier is a simpler method. /IMAGE creates a functionally equivalent copy of the input volume on the output volume. The Backup utility copies the information to the output disk in a contiguous manner, compressing the files, so a large block of unused space is available on the output volume. This compression usually improves the performance of the volume. Therefore, image backups are highly recommended. Table 4–21 shows how to use the /IMAGE qualifier to back up volumes to tape or disk and to restore them again.

Table 4–21 Backing Up and Restoring Image Volumes

Command	Function
Saving a disk volume to a save set on tape	

`$ BACKUP/IMAGE -`
`_$ device-name -`
`_$ save-set-specifier`
	Creates a save set on tape so you can restore data if failures occur, or so you can temporarily store data on a less expensive medium. Tape volumes must be mounted with the /FOREIGN qualifier first.

`$ BACKUP/IMAGE -`
`_$ DBA1: -`
`_$ MUA0:MAR24.BCK`	

`$ BACKUP/IMAGE/RECORD -`
`_$ DBA1: -`
`_$ MUA0:FULL.BCK`
	The /RECORD qualifier causes the utility to record the current date and time in the backup field of the file header of every file on DBA1 as it saves them. The Backup utility checks this field when it does an incremental backup (see Table 4–22).

Restoring a disk volume from an image save set	

`$ BACKUP/IMAGE -`
`_$ save-set-specifier -`
`_$ device-name`
	Restores a save set after a failure occurs and the data on the original disk volume is destroyed, or moves data from temporary storage back to a disk volume.

`$ BACKUP/IMAGE -`
`_$ MUA0:MAR24.BCK -`
`_$ DBA1:`
	Copies the contents of the MUA0:MAR24.BCK save set to the disk volume, DBA1. Since you include the /IMAGE qualifier, BACKUP creates the directories and subdirectories named in the save set to duplicate the original volume on the output volume.

Copying a disk volume to another disk and creating a more efficient volume	

`$ BACKUP/IMAGE -`
`_$ input-device -`
`_$ output-device`
	Makes a copy of the volume loaded in the input device to the volume loaded in the output device.

`$ BACKUP/IMAGE -`
`_$ DBA1: -`
`_$ DBA2:`
	Copies the contents of the volume in device DBA1 to the volume in device DBA2. In the new volume, the files are made contiguous, making I/O more efficient.

Save the entire contents of a volume on a weekly or monthly basis. In between full image backups, do **incremental** backups. An incremental backup saves only those files created or modified since the last backup where the /RECORD qualifier was used. Always restore incremental backups in reverse order, including the /INCREMENTAL qualifier (see Table 4–22).

Table 4–22 Incremental Backup and Restore

Command	Function
Incremental Backup	

`$ BACKUP/RECORD -`
`_$ device-name/SINCE=BACKUP -`
`_$ save-set-specifier`
	Backs up those files that were created or modified since the last save operation. BACKUP determines if a file was created or modified since the last backup by comparing the backup field in the file header with the modified field. The /RECORD qualifier causes the utility to write the current date and time to the file header of each file it saves.

`$ BACKUP/RECORD -`
`_$ DRA1:[*...]/SINCE=BACKUP -`
`_$ MUA0:JUN19F`
	Copies all files in all directories on the DRA1 volume that were created or done with the /RECORD qualifier. Files are copied to the save set MUA0:JUN19F. You must mount the volume with the /FOREIGN qualifier before entering this command.

Incremental Restore	

`$ BACKUP/INCREMENTAL -`
`_$ save-set-specifier -`
`_$ output-device -`
`_$ /BY_OWNER=ORIGINAL`
	Restores incremental backup save sets in reverse order (i.e., restores the save set made on Friday, then Thursday's, then Wednesday's, etc.) after you restore the most recent full image save set. Include the /INCREMENTAL qualifier when restoring incremental save sets to enable the utility to restore the files correctly. Include the /BY_OWNER=ORIGINAL qualifier to ensure that each file is given the UIC of its original owner, rather than the UIC of the process entering the BACKUP command.

`$ BACKUP/IMAGE -`
`_$ MUA0:WEDFULL.BCK -`
`_$ DRA1:`
 	
`$ BACKUP/INCREMENTAL -`
`_$ MUA0:FRIINC.BCK -`
`_$ DRA1:[*...]/BY_OWNER=ORIGINAL`
`$ BACKUP/INCREMENTAL -`
`_$ MUA0:THUINC.BCK -`
`_$ DRA1:[*...]/BY_OWNER=ORIGINAL`
	Restores the full image backup done on Wednesday to the device DRA1. The incremental save sets update the volume until it returns to the same state it was in at the time of the last incremental backup (done on Friday).

4.15 Laboratory Exercises

You need a tape drive and tape volume to complete these exercises.

1. Create three subdirectories of your default login directory. Call them [.A], [.B], and [.A.A2].
2. Copy at least two files into each of the above three subdirectories.
3. Allocate a tape drive. (If all drives are already allocated by other users, return to this exercise later.)

   Load your scratch tape into the drive. If you do not know how to load this drive, consult your course administrator.
4. Initialize the scratch tape, then mount the tape as a foreign volume. Back up all the files in your directory and its subdirectories to a single save set on the tape.
5. List the contents of your new save set.
6. Delete one file in [.B] and selectively restore that file from your save set on tape.
7. Delete all files in [.A] and [.A.A2] and the subdirectory files themselves. Issue the following DCL commands:

   ```
   $ DELETE [username.A]*.*;*,[username.A.A2]*.*;*
   $ BACKUP  tape:save_set_name  disk:[username...]
   ```

   Note the results.

   What command would you use to correctly restore your subdirectories?
8. Dismount the tape drive. Issue a REPLY command to inform users that the tape drive is now available.

4.16 Solutions to Laboratory Exercises

Substitute your own directory and device names for the ones shown in these solutions. You can also choose your own tape label and your own file name for the save set.

1.
```
$ CREATE/DIRECTORY [SYSMn.A]
$ CREATE/DIRECTORY [SYSMn.B]
$ CREATE/DIRECTORY [SYSMn.A.A2]
```

2.
```
$ COPY SYS$LOGIN:*.* [.A]
$ COPY SYS$LOGIN:*.* [.B]
$ COPY SYS$LOGIN:*.* [.A.A2]
```

3.
```
$ ALLOCATE WHOOSH$MUA0:
```

4.
```
$ INITIALIZE WHOOSH$MUA0: MYTAPE
$ MOUNT/FOREIGN WHOOSH$MUA0:
$ BACKUP [...]*.*;* WHOOSH$MUA0:MYFILES.BCK /LABEL=MYTAPE
```

5.
```
$ BACKUP /REWIND /LIST WHOOSH$MUA0:MYFILES.BCK
```

6.
```
$ DELETE [MYNAME.B]LOGIN.COM;*
$ BACKUP/LOG WHOOSH$MUA0:MYFILES.BCK/SELECT=[MYNAME.B]LOGIN.COM -
_$ [MYNAME.B]
```

7. The commands shown cause BACKUP to create an extra level of subdirectories. The correct way to restore all your files is to use a command like the following:

```
$ BACKUP  WHOOSH$MUA0:MYFILES.BCK  STUDENT$DISK:[*...]
```

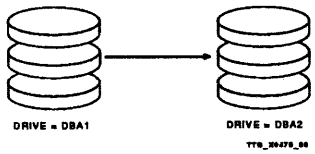
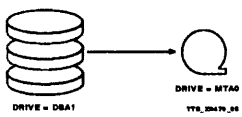
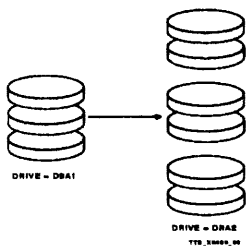
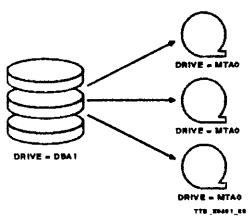
8.
```
$ DISMOUNT WHOOSH$MUA0:
$ DEALLOCATE WHOOSH$MUA0:
$ REPLY/USERS "Tape drive WHOOSH$MUA0 is now available"
```

4.16.1 Operator Action During Multivolume Disk or Tape Set Backups

Backups can be made to disk or to tape. Sometimes the information on an input disk fits on one piece of output media. At other times, multiple pieces of output media must be used to contain all the data (see Figure 4-4). These are referred to as multivolume disk or tape sets.

An operator who receives a request to load the next volume of a multivolume disk or tape set should:

1. Unload the current volume.
2. Load the next volume.
3. Respond to the request using the REPLY command with the /INITIALIZE or /BLANK TAPE qualifiers (see Example 4–7).

OPERATION/COMMAND	COMMENTS
Backup Disk to Disk - $ BACKUP/IMAGE DBA1 DBA2  	Copy all of the information from the input disk to the output disk. If the output disk is large enough to hold all of the information from the input disk, you can store the input disk and use the output disk. Performance improves because the information on the output disk is compressed and is faster to access. Note that if the input disk was bootable, the output disk will also be bootable.
Backup Disk to Tape - $ BACKUP/IMAGE DBA1 MTA0:FULL.BCK  	Copy all of the information from the disk to a save set on the tape. To improve performance, load a scratch disk and restore the information to it from the tape. Then use the newly created disk and store the previous one.
Backup Disk to Multiple Disks - $ BACKUP/IMAGE DBA1: - _$ DRA2:SAVE.BCK  	Create a save set on the output volumes. When the first output volume is full, the system notifies the operator by sending a message to the operator console. The operator removes the volume and loads the next one. The operator is prompted to load each successive volume until the backup is complete. Improve performance with the same method used in disk to tape backups (see previous section of figure). These output disks are called a multivolume disk set. To restore them, you must begin with the first volume.
Backup Disk to Multiple Tapes - $ BACKUP/IMAGE DBA1: - _$ MTA0:SAVE.BCK  	Create a save set. When the first volume is full, the system notifies the operator to load the next volume. This continues until the backup is complete. Improve performance using the same method described for disk to tape backups (See previous section of figure). The output tapes are called a multivolume tape set. To restore them, you must begin with the first volume.

TTB_X0971_88

Figure 4–4 Using Output Media for Backups

```
❶ $ ALLOCATE MU: TAPE
  %DCL-I-ALLOC, _MUA0: allocated
  $ INITIALIZE TAPE BACK1
  $ MOUNT TAPE BACK1
  %MOUNT-I-MOUNTED, BACK1 mounted on _MUA0:
❷ $ SHOW DEVICE/FULL TAPE

  Magtape MUA0:, device type TK50, is online, allocated, mounted, file-
          oriented device, error logging is enabled.

      Error count                    0     Operations completed             25
      Owner process       "Chocoholic"     Owner UIC                   [11,340]
      Owner process ID       20C000AF      Dev Prot S:RWED,O:RWED,G:RWED.W:RWED
      Reference count                1     Default buffer size            2048

      Volume label           "BACK1 "      Relative volume no.               1
      Record size                    0     Transaction count                 1
      Mount status             Process     Mount count                       1
      ACP process name      "MUA0CACP"
      Density              833 (normal)    Format                    Normal-11

    Volume status:  beginning-of-tape, odd parity.

❸ $ COPY SYS$SYSTEM:SYSDUMP.DMP TAPE
  $ COPY/EXCLUDE=*.DAT   SYS$LIBRARY:*.*  TAPE

  %%%%%%%%%%  OPCOM  4-MAY-1989  14:35:20.47  %%%%%%%%%%
  Request 1, from user BIERLY
  MOUNT new relative volume 2 (BACK1_2) on_MUA0:

❹ $ SET PROCESS/PRIVILEGE=OPER
  $ REPLY/INITIALIZE=1 "BACK2"
  BACK2
  14:41:16.86, request 1 was completed by operator _OPA0:

❺ $ SHOW DEVICE/FULL TAPE

  Magtape MUA0:, device type TK50, is online, allocated, mounted, file-
          oriented device, error logging is enabled.

      Error count                    3  Operations completed            14110
      Owner process       "Chocoholic"  Owner UIC                   [11,340]
      Owner process ID       20C000AF   Dev Prot  S:RWED,O:RWED,G:RWED.W:RWED
      Reference count                1  Default buffer size            2048

      Volume label           "BACK2 "   Relative volume no.               2
      Record size                    0  Transaction count                 1
      Mount status             Process  Mount count                       1
      ACP process name      "MUA0CACP"
      Density              833 (normal)  Format                    Normal-11

    Volume status:  end-of-file, odd parity.

❻ $ DISMOUNT TAPE
  $ DEALLOCATE TAPE
  $ DEASSIGN TAPE
```

Example 4–7 Creating a Multivolume Tape Set on One Drive

Notes on Example 4–7:

This example shows how the operating system automatically intervenes to create a multivolume tape set. It does this when you attempt to write a file to a tape volume that does not have sufficient space.

❶
```
$ ALLOCATE MU: TAPE
$ INITIALIZE TAPE BACK1
$ MOUNT TAPE BACK1
```

To prepare a tape volume, allocate a device, physically load the tape reel, initialize it, and mount it. The device name MUA0 is assigned to the logical name TAPE. The assignment is recorded in your process logical name table.

❷
```
$ SHOW DEVICE/FULL TAPE
```

The SHOW DEVICE/FULL command displays the characteristics of your newly created tape volume. In this example, you are copying information to a set of tapes. The MOUNT command assigns a relative volume number to each tape you load and includes this number in informational or error messages to the console. The value in the relative volume number field for this first tape is zero. These are internal numbers used by the system; they are not recorded on the tape volumes. Therefore, attach a label to each tape you create specifying its position in the set (first tape, second tape, third tape, etc.). Then, when you use the set later, you can load the tapes in the proper order.

❸
```
$ COPY SYS$SYSTEM:SYSDUMP.DMP TAPE
$ COPY/EXCLUDE=*.DAT SYS$LIBRARY:*.* TAPE
```

These two COPY commands cause selected files to be copied from system directories to TAPE. Each file appears on the tape in the order in which it appears in each of the specified directories. The group of files specified by the second COPY command string does not fit on BACK1. As a result, the system temporarily suspends the operation and sends a message to the single operator terminal on the system.

The message includes the system-generated name of BACK1_2 for the second tape. The system creates the name of subsequent tapes in a multivolume set from the name of the first volume (unless you specify all the names of the volumes in the first MOUNT command).

❹
```
$ REPLY/INITIALIZE=1 "BACK2"
```

Some events that occur on a VMS system require intervention by an operator. In the absence of an operator, you must log in to the system at an operator terminal and issue the necessary commands (you need OPER privilege to do this). In this listing, the boxed section contains the dialogue seen on the operator's terminal.

The message sent to the operator terminal by the system includes the name of your process, USER, and a unique identification number. The operator uses this number to respond to the request (/INITIALIZE=1.)

Before the operator responds to the request, he or she unloads the full tape and loads either a blank tape volume or a volume you own on the drive. The operator then responds to the message using the REPLY command to initialize the new volume and signal the system to continue the interrupted copy operation:

```
$ REPLY/INITIALIZE=1 "BACK2"
```

In this example, the operator specifies BACK2 as the label of the second volume. This does not match the default name that the system would assign because the operator is not restricted to the default name. (Regardless of the name used, the operator must precede it with a quotation mark.)

The system issues a completion message to the operator terminal when it has successfully mounted the volume.

❺
```
$ SHOW DEVICE/FULL TAPE
```

Returning to your own terminal, note that no messages have been sent to your process. There is no indication on your terminal that the system had to take any special action to complete execution of your final COPY command. However, on systems with disk and tape operators, a member of the operations staff will typically send you an explicit message.

Once the second volume is properly installed on MUA0, the copy operation completes and the VMS operating system displays a DCL dollar sign prompt ($) at your terminal. The SHOW DEVICE command reveals that the volume you mounted, BACK1, is no longer present on MUA0. Instead, MUA0 now contains BACK2. The relative volume number of BACK2 is two (2). The protection code assigned to the volume is identical to the one you specified for BACK1.

❻
```
$ DISMOUNT TAPE
$ DEALLOCATE TAPE
$ DEASSIGN TAPE
```

When you have completed your work with BACK2, dismount and unload the tape. Once you have physically removed the volume from MUA0, deallocate the device. The DEASSIGN command deletes the logical name TAPE from your process logical name table.

4.16.2 Creating Multivolume Tape Sets on More than One Drive

If you have more than one tape drive on your system, you can request the system to automatically load tapes in a multivolume tape set. To implement this feature, list the names of the drives in the MOUNT command.

If you are creating a multivolume tape set, and you include the /INITIALIZE=CONTINUATION qualifier with the MOUNT command, the system will initialize and mount tapes on the drives listed in the command. It will also generate label names (if you did not provide them in the INITIALIZE or MOUNT commands) and write them on the tapes.

If you are reading a multivolume tape set, and you do not list the volume labels with the MOUNT command, the system generates a label for each volume and compares it to the label on that volume. If they match, the system will process that tape volume.

The system always selects the next drive on which to process a volume from the list of drives specified in the MOUNT command. For example:

```
$MOUNT MUA0:,MUA1:,MUA2: TAPE
```

The system processes the TAPE volume on the MUA0: drive, followed by the volumes on MUA1: and MUA2: respectively. If the set contains more volumes, the fourth volume is loaded on MUA0:, the fifth on MUA1:, and so on.

The system processes tape volumes one at a time. So, while the system is processing the third tape in a set on drive MUA2:, you can be changing the volumes on MUA0: and MUA1: respectively. The system sends a message to the operator's console when it begins processing a new tape. By reading these messages, you know when the system has completed the previous tape so you can remove, label, and store it.

To generate labels, the system combines the first four characters of the label you specify with the relative volume number (using two digits to represent this number). If your label is shorter than four characters, the system includes the underscore character (_) between the characters and the number.

Table 4–23 shows the types of labels the system generates based on the label you specify.

Table 4–23 Generating Labels Automatically

Specified Label (First Volume)	Corresponding Generated Label (Second and Subsequent Volumes)
MAIN	MAIN02, MAIN03,... MAIN99
T15	T15_02, T15_03,... T15_99
DAN	DAN_02, DAN_03,... DAN_99
BACKUP	BACK02, BACK03,... BACK99

4.16.3 BACKUP Tape Label Processing

When a tape is initialized, an expiration date is written into the tape header. The purpose of the expiration date is to keep the tape from being prematurely or accidentally overwritten.

Before writing to a tape, BACKUP first checks the tape label to see if it has expired. If the current date is earlier than the expiration date, BACKUP issues an error message and automatically unloads the tape from the drive.

To specify an expiration date on a newly created backup tape, specify the desired date with the /TAPE_EXPIRATION qualifier:

```
$ BACKUP /RECORD/SINCE=YESTERDAY  DISK$USER:[*...] -
_$ MUA0:MAY0688.BCK /REWIND/TAPE_EXPIRATION=30-MAY-1988
```

To overwrite a tape that has not yet expired, use the /IGNORE=LABEL_PROCESSING qualifier (see Table 4–14). If you override tape label processing by using this qualifier, make sure you properly set the new expiration date with the /TAPE_EXPIRATION qualifier.

Besides volume protection, expiration dates are the only form of protection against overwriting data on a tape. Since privileged accounts usually perform BACKUP operations, the volume protection does not ensure against accidental overwriting of a tape. Proper, consistent use of the /TAPE_EXPIRATION qualifier should be your first defense against such situations.

4.16.4 BACKUP Journal Files

If you manage a large or very active system on a regular basis, your backup save sets will typically span more than one tape volume. This can become a problem when trying to locate a file that must be restored from one of the tapes in a multivolume save set.

When restoring a file from a save set, issue the appropriate BACKUP restore command, then mount the first tape in the multivolume save set. BACKUP reads the file information on tape, looking for the specified file. If it reaches the end of the tape, it issues a message instructing you to load the next tape in the save set, and unload the current tape from the drive. BACKUP continues in this manner until it either finds the file, or reaches the end of the multivolume save set (on the last tape). This procedure can often take a considerable amount of time.

BACKUP provides a facility called **journal files** to help you manage your tape save sets, especially for file restorations. When you issue a BACKUP save command, you can have BACKUP create a journal file for the save set. A journal file contains all volume, directory, and file information pertaining to the files specified in the BACKUP save command.

To create a journal file when performing a BACKUP save operation, include the /JOURNAL=file-spec qualifier in the BACKUP command:

```
$ BACKUP /RECORD/JOURNAL=MAY0688.BJL -
_$ DISK$USER:[PAYROLL...] -
_$ MUA1:MAY0688.BCK /REWIND/TAPE_EXPIRATION=13-MAY-1988
```

In this example, a journal file called MAY0688.BJL is created as BACKUP saves the appropriate files in the DISK$USER:[PAYROLL...] directory tree. (BACKUP uses a default extension of .BJL if you omit the file extension when specifying the /JOURNAL=file-spec qualifier.)

Examine the contents of the journal file by issuing a BACKUP command with the qualifiers /LIST/JOURNAL=file-spec, for example:

```
$ BACKUP /LIST/JOURNAL=MAY0688.BJL
```

No other qualifiers or parameters should be specified when listing a journal file.

> NOTE
> A BACKUP journal file is written as a specially formatted binary file. Thus, you cannot simply print or edit a journal file to examine its contents. Use only the BACKUP command shown above to list the contents of a BACKUP journal file.

If you have created a journal file and later need to restore a particular file from the associated tape save set, list the contents of the journal file and look for the location of the file. Next issue the appropriate BACKUP restore command (see Table 4-24), but instead of loading the first tape in the multivolume save set, mount the volume containing the desired file.

Many sites make the creation of journal files an integral part of regular disk volume save operations. The journal files are stored in an appropriate subdirectory owned by a privileged account.

Table 4–24 Common BACKUP Operations

Operation	Save Command	Restore Command
Full backup operations		
Physical backup	`$ BACKUP/PHYSICAL DUA0: DUA1:`	

Image, disk to tape	`$ BACKUP/IMAGE/RECORD -`
	`_$ DUA0: -`
	`_$ MUA0:01JAN1990.BCK-`
	`_$ /INITIALIZE/REWIND/BUFFER=5 -`
	`_$ /TAPE_EXPIRATION=8-JAN-1990 -`
	`_$ /DENSITY=1600/BLOCK=32768`
		`$ BACKUP/IMAGE -`
		`_$ MUA0:01JAN1990.BCK/BUFFER=5 -`
		`_$ DUA0:`

Image, disk to disk	`$ BACKUP/IMAGE/RECORD -`
	`_$ DUA0: -`
	`_$ DUA1:`	

Image, disk to save set in disk directory	`$ BACKUP/IMAGE/RECORD -`
	`_$ DUA0: -`
	`_$ DUA1:[BACKUP]01JAN1990.BCK -`
	`_$ /SAVE_SET`
		`$ BACKUP/IMAGE -`
		`_$ DUA1:[BACKUP]01JAN1990.BCK -`
		`_$ /SAVE_SET -`
		`_$ DUA0:`

Image, disk to save set on multiple disks	`$ BACKUP/IMAGE/RECORD -`
	`_$ DUA0: -`
	`_$ DUA1:01JAN1990.BCK/SAVE_SET`
		`$ BACKUP/IMAGE -`
		`_$ DUA1:01JAN1990.BCK/SAVE_SET -`
		`_$ DUA0:`

Table 4–24 (Cont.) Common BACKUP Operations

Operation	Save Command	Restore Command
Incremental Backup Operations		

Disk to tape	`$ BACKUP/SINCE=BACKUP/RECORD -`
	`_$ DUA0:[000000...]*.*.* -`
	`_$ MUA0:01JAN1990.BCK -`
	`_$ /INITIALIZE/REWIND/BUFFER=5 -`
	`_$ /TAPE_EXPIRATION=8-JAN-1990 -`
	`_$ /BLOCK=32768/DENSITY=1600`
		`$ BACKUP/INCREMENTAL -`
		`_$ MUA0:01JAN1990.BCK/BUFFER=5 -`
		`_$ DUA0:`

Disk to save set in disk directory	`$ BACKUP/SINCE=BACKUP/RECORD -`
	`_$ DUA0:[000000...]*.*.* -`
	`_$ DUA1:[BACKUP]01JAN1990.BCK -`
	`_$ /SAVE_SET`
		`$ BACKUP/INCREMENTAL -`
		`_$ DUA1:[BACKUP]01JAN1990.BCK -`
		`_$ /SAVE_SET -`
		`_$ DUA0:`

Disk to save set on multiple disks	`$ BACKUP/SINCE=BACKUP/RECORD -`
	`_$ DUA0:[000000...]*.*.* -`
	`_$ DUA1:01JAN1990.BCK/SAVE_SET`
		`$ BACKUP/INCREMENTAL -`
		`_$ DUA1:01JAN1990.BCK/SAVE_SET -`
		`_$ DUA0:`

Partial Backup Operations		

Disk to tape	`$ BACKUP/RECORD -`
	`_$ DUA0:[SMITH...]*.*.* -`
	`_$ MUA0:SMITH.BCK -`
	`_$ /INITIALIZE/REWIND/BUFFER=5 -`
	`_$ /TAPE_EXPIRATION=1-JAN-1993 -`
	`_$ /DENSITY=1600/BLOCK=32768`
		`$ BACKUP -`
		`_$ MUA0:SMITH.BCK/BUFFER=5 -`
		`_$ DUA0:`

Disk to disk	`$ BACKUP -`
	`_$ DUA0:[SMITH...]*.*.* -`
	`_$ DUA1:`	

Disk to save set in disk directory	`$ BACKUP/RECORD -`
	`_$ DUA0:[SMITH...]*.*.* -`
	`_$ DUA1:[BACKUP]SMITH.BCK -`
	`_$ /SAVE_SET`
		`$ BACKUP -`
		`_$ DUA1:[BACKUP]SMITH.BCK -`
		`_$ /SAVE_SET -`
		`_$ DUA0:`

Disk to save set on multiple disk	`$ BACKUP/RECORD -`
	`_$ DUA0:[SMITH...]*.*.* -`
	`_$ DUA1:SMITH.BCK/SAVE_SET`
		`$ BACKUP -`
		`_$ DUA1:SMITH.BCK/SAVE_SET -`
		`_$ DUA0:`

4.16.5 Standalone BACKUP

VMS system utilities and programs need information on the system disk to run. The system stores temporary information on the system disk while managing the users. As a result, the system disk is continually being used while the VMS operating system is running. The discussion of on-line BACKUP stated that no one should be using a disk while the Backup utility is copying it. Therefore, you must shut down the VMS operating system before attempting to back up the system disk. Module 6 discusses how to shut down your system.

To copy your system disk, boot your standalone BACKUP kit. (Boot procedures differ from processor to processor. Refer to the section on standalone utilities and diagnostics in Module 6 for an example and further discussion of this.) The standalone BACKUP kit contains enough VMS system software to produce a DCL prompt and to display error messages. The kit also includes enough of the Backup utility software to copy the system disk and do some other basic operations. Always have at least two working copies of this kit available. You can produce copies of the kit by executing the SYS$UPDATE:STABACKIT.COM procedure.

After you have completed the boot procedure, VMS system software displays a DCL prompt at your console terminal. At this point, enter a BACKUP/IMAGE command. Be careful not to generate an error or the console medium will be searched for an appropriate error message. This search can be time consuming, especially when the console medium is a TU58.

Back up the system disk to another disk or to a tape (see Table 4–24). When the backup is complete, reload the console medium and boot the system.

Module 7 contains an example of the standalone BACKUP kit. Examples of standalone BACKUP can also be found in the *VMS Backup Utility Manual.*

4.17 Verify Utility

The Verify utility checks the readability and validity of the file structure. VERIFY allows the following modes of operation:

- Error reporting with no repairs
- Error reporting with repairs
- Error reporting with user-controlled selective repairs

The command format for invoking the Verify utility is:

```
$ ANALYZE/DISK_STRUCTURE device-name [/qualifier...]
```

Table 4–25 and Example 4-8 give examples of the Verify utility.

Execute the utility once **without** the /REPAIR qualifier (to locate errors and determine corrective action) and again using the /REPAIR or /REPAIR/CONFIRM qualifiers (to make needed repairs). To run VERIFY on a system disk, force all users off the system before executing the Verify utility. (Module 6 discusses system shutdown procedures in more detail.) For the list of error messages that the Verify utility issues, read the section on error messages in the *VMS Analyze/Disk_Structure Utility Manual.*

Table 4–25 Using the Verify Utility

Function	Command	Comments

Running VERIFY to find errors in the file structure	`$ ANALYZE/DISK_STRUCTURE -`
	`_$ device-name`
	
	`$ ANALYZE/DISK_STRUCTURE -`
	`_$ MYDISK`
		Using no parameter qualifier invokes VERIFY in the error-reporting mode.

Running VERIFY to report and repair errors	`$ ANALYZE/DISK_STRUCTURE -`
	`_$ device-name/REPAIR`
	
	`$ ANALYZE/DISK_STRUCTURE -`
	`_$ MYDISK/REPAIR`
		The /REPAIR qualifier causes automatic repair of all errors located in the file structure.

Running VERIFY to report errors and allow the user to decide what gets repaired	`$ ANALYZE/DISK_STRUCTURE -`
	`_$ device-name/REPAIR/CONFIRM`
	
	`$ ANALYZE/DISK_STRUCTURE -`
	`_$ MYDISK/REPAIR/CONFIRM`
		The qualifiers /REPAIR and /CONFIRM cause the utility to display errors, ask the user if the error should be corrected, and carry out the user's response.

Example 4-8 shows how the Verify utility analyzes the structure of a private disk.

```
$ MOUNT DRA2 PROGRAM_DISK MYDISK
%MOUNT-I-MOUNTED, PROGRAM_DISK mounted on _DRA2:
❶ $ ANALYZE/DISK_STRUCTURE MYDISK
%VERIFY-I-OPENQUOTA, error opening QUOTA.SYS
-SYSTEM-W-NOSUCHFILE, no such file
%VERIFY-I-LOSTHEADER, file (11,2,1) UTLIST.DAT;1
        not found in a directory
%VERIFY-I-LOSTHEADER, file (12,2,1) UTLIST.COM;1
        not found in a directory
$
❷ $ ANALYZE/DISK_STRUCTURE/REPAIR/CONFIRM MYDISK
%VERIFY-I-OPENQUOTA, error opening QUOTA.SYS
-SYSTEM-W-NOSUCHFILE, no such file
%VERIFY-I-LOSTHEADER, file (11,2,1) UTLIST.DAT;1
❸       not found in a directory
Repair this error (D to delete)? (D, Y or N):  Y
%VERIFY-I-LOSTHEADER, file (12,2,1) UTLIST.COM;1
        not found in a directory
Repair this error (D to delete)? (D, Y or N):  Y
$
❹ $ DIRECTORY MYDISK:[SYSLOST]

Directory MYDISK:[SYSLOST]

UTLIST.COM;1        UTLTST.DAT;1

Total of 2 files.
$
```

Example 4–8 Using the Verify Utility

Notes on Example 4–8:

❶ `$ ANALYZE/DISK_STRUCTURE MYDISK`

VERIFY is run in the diagnostic mode to find faults. To help decide on a course of action, read the section on VERIFY messages in the *VMS Analyze/Disk_Structure Utility Manual.*

❷ `$ ANALYZE/DISK_STRUCTURE/REPAIR/CONFIRM MYDISK`

After deciding to recover the lost files, run VERIFY again, this time with the /REPAIR/CONFIRM qualifiers, to make selective repairs.

❸ `% VERIFY-I-LOSTHEADER, ...`

The Verify utility asks you which of the following actions to take:

- Recover the file.
- Delete the file.
- Let the file remain lost. In this case, request that the file be recovered by typing Y. Recovered files are placed in the directory [SYSLOST].

❹ `$ DIRECTORY MYDISK:[SYSLOST]`

This command lists all "lost" files on the disk MYDISK.

4.18 Transferring Files Between VAX and PDP-11 Systems

By using the VMS Exchange utility or the RMS utilities RMSBCK and RMSRST, you can transfer files between the VMS operating system and other systems. RMS files on other systems can be transferred to tape using RMSBCK, and restored from tape to the target system using RMSRST. Transfer non-RMS files with the Exchange utility. Table 4–26 describes how to transfer files between VAX and other systems.

4.18.0.1 Additional References

- To learn more about what to do to maintain private and public volumes, consult the *Guide to Maintaining a VMS System*.
- To learn more about how to use the on-line Backup utility, consult the *VMS Backup Utility Manual*.
- To learn more about how to use the Verify utility, consult the *VMS Analyze/Disk_Structure Utility Manual*.
- For more information on the various utilities you can use to transfer files, consult the *VMS Exchange Utility Manual*.

Table 4–26 Transferring Files Between VAX and PDP-11 Systems

System	Operation	Comments
RSX	Preparing a volume on the VMS operating system for use on RSX	The only special action needed to prepare a volume for use on RSX is to initialize the volume using the /STRUCTURE_LEVEL=1 qualifier. See the description of the INITIALIZE command in the *VMS DCL Dictionary*.
	Preparing a volume on RSX for use on the VMS operating system	VMS understands the RSX file structure. You can create RSX volumes and use them on a VMS system without taking special action.
RSTS	Transferring RMS files between systems	Use RMSBCK to create a tape volume containing the files you want to transfer. Transfer the files from the tape to the target system using RMSRST. See the the *RMS-11 User's Guide* for more information.
	Transferring non-RMS files between systems	Use the Exchange utility to transfer the files from the transportable medium to the target system. See the *VMS Exchange Utility Manual* for more information.
RT-11	Transferring files between RT-11 and the VMS operating system	Use the Exchange utility. See the *VMS Exchange Utility Manual*.