Educational Services

VMS System and Network Management II: Managing Established Systems

Student Workbook

EY-G987E-SG-0001

Digital Equipment Corporation

CONTENTS

4 Performing Backups and Restores

5 Introduction to System Customization

6 Layered Product Installation

10 System Monitoring

11 Developing Command Procedures

12 Written Exercises

13 Laboratory Exercises

14 Test

INDEX

EXAMPLES

FIGURES

TABLES

About This Course

INTRODUCTION

The *VMS System and Network Management II* course is designed to give system managers of VMS systems and networks additional information about how to manage a computer running the VMS operating system.

This student workbook is divided into a number of chapters, each designed to cover a well-organized topic, or group of topics. Most chapters include figures, tables, and examples to enable students to better understand the material. Two separate exercise chapters (one for written exercises and one for laboratory exercises) can be found at the back of this workbook to allow students to test their VMS system and network management skills.

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

- Course Overview
- Intended Audience
- Prerequisites
- Course Goals
- Course Nongoals
- Course Organization
- Course Map
- Resources
- Course Conventions

COURSE OVERVIEW

This task-oriented course continues the system management training presented in *VMS System and Network Management I*. It prepares the participant to perform and automate daily system management tasks in an existing VMS or VAXcluster system and network environment. This course provides students with the information and guided practice they need to perform the tasks required of system managers in an operational environment, such as maintaining disks and queues, performing full and incremental backups, and installing layered products. Coverage of the use of command procedures to automate repetitive tasks is included. The course also introduces the strategies and methods used to secure a VMS environment as well as the tools for monitoring system activity.

Each chapter builds on the topics covered in the *VMS System and Network Management I* course and also on the individual student's system and network management experience.

The *VMS System and Network Management II* course covers:

- Managing Disks
- Using Logical Name Tables
- Queue Management
- Performing Backups and Restores
- Introduction to System Customization
- Layered Product Installation
- Reporting on User Activity
- Maintaining System Security
- System Monitoring
- Managing a Network Node
- Developing Command Procedures

INTENDED AUDIENCE

This course is for system managers who require basic system, cluster, and network management skills. These system managers have taken *VMS System and Network Management I* or have similar knowledge and experience, and are ready to learn how to perform the full complement of daily system management and operations tasks.

The course assumes that the students are to manage systems that have already been installed and customized, and that on returning to their sites, each student has access to at least one experienced system manager who can perform system customization and provide support.

PREREQUISITES

Before taking this course, the system manager should be able to:

- Perform basic user tasks on a VMS system, including:
  - Logging in and out
  - Sending mail messages to other users
  - Editing a text file with a text editor
- Manage system users, which requires:
  - Maintaining the user authorization file and volume quota files
  - Creating user file directories (UFDs)
  - Controlling user processes
- Manage system resources, which requires:
  - Mounting and dismounting disk and tape volumes
  - Setting device characteristics
  - Starting and stopping print and batch queues
  - Performing full backup and restore on a disk volume
  - Starting up and shutting down a system using the default startup procedure

These prerequisites can be satisfied by taking the *VMS System and Network Management I* course or obtained through experience operating or managing a VMS system.

COURSE GOALS

To perform daily system management, the system manager should be able to:

- Manage queues, disks and tapes, and terminals
- Perform all types of backup and restore operations
- Describe the functions of the system startup and login files
- Use AUTOGEN and SYSMAN to set system parameters
- Install VMS layered products and updates
- Use Accounting to collect process information
- Describe general system security mechanisms
- Monitor the system for certain behavioral problems
- Describe general tasks for managing a network node
- Write and use command procedures to automate system management tasks

NONGOALS

VMS System and Network Management II does **not** cover the following topics:

- Basic use of a VMS system (covered in *VMS System and Network Management I*)
- System installation, including VMS operating system installation, adding nodes to a VAXcluster system, or adding nodes to a network (covered in *VMS System and Network Management III*)
- System customization, including customizing boot procedures (covered in *VMS System and Network Management III*)
- System performance management and tuning (introduced in *VMS System and Network Management III* and taught in *VMS System Performance Management*)
- Details of system security features (taught in *VMS System Security Features*)
- System troubleshooting

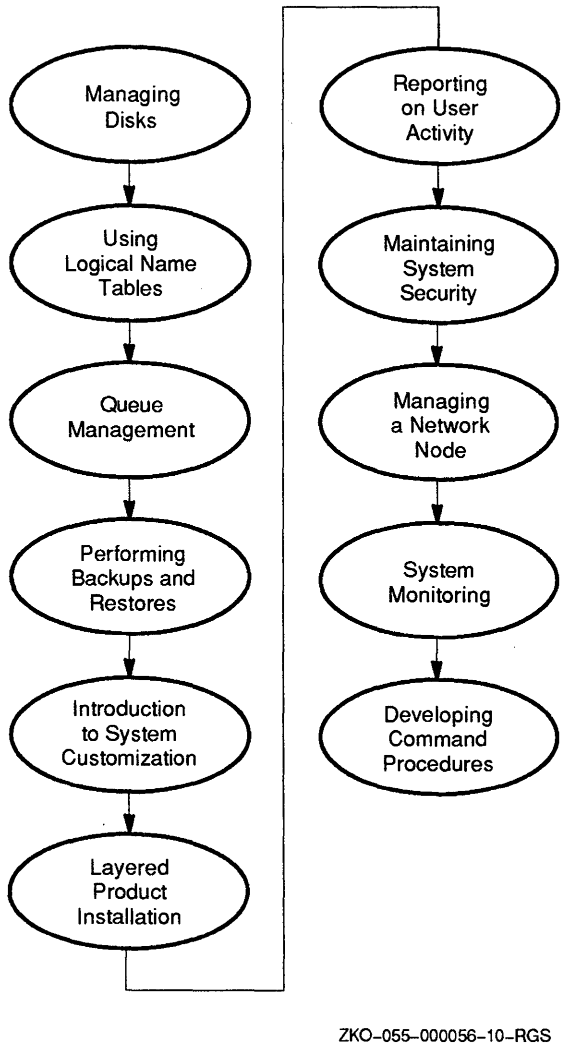
COURSE ORGANIZATION

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

- An **introduction**, which describes the purpose of the chapter, provides motivation for mastering its objectives, and outlines its contents.
- One or more **objectives**, which identify the skills taught in the chapter. Objectives are designed to focus your study efforts on a selected number of skills.
- The chapter **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 chapter **summary** that reviews important concepts and skills taught in the chapter

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

COURSE MAP



Managing Disks
Using Logical Name Tables
Queue Management
Performing Backups and Restores
Introduction to System Customization
Layered Product Installation
Reporting on User Activity
Maintaining System Security
Managing a Network Node
System Monitoring
Developing Command Procedures
ZKO-055-000056-10-RGS

RESOURCES

The books and manuals listed here should be available for your reference in the classroom or in the laboratory.

- *Guide to DECnet–VAX Networking*
- *Guide to Maintaining a VMS System*
- *Guide to Setting Up a VMS System*
- *Guide to Using VMS*
- *Guide to Using VMS Command Procedures*
- *Guide to VMS File Applications*
- *Guide to VMS Files and Devices*
- *Guide to VMS Performance Management*
- *Guide to VMS System Security*
- *Introduction to VMS System Management*
- *VAX Systems and DECsystems Systems and Options Catalog*
- *VMS Accounting Utility Manual*
- *VMS Analyze/Disk_Structure Utility Manual*
- *VMS Audit Analysis Utility Manual*
- *VMS Authorize Utility Manual*
- *VMS Backup Utility Manual*
- *VMS DCL Concepts Manual*
- *VMS DCL Dictionary*
- *VMS Install Utility Manual*
- *VMS Guide to Disk and Magnetic Tape Operations*
- *VMS Installation and Operations* guides
- *VMS License Management Utility Manual*

- *VMS Monitor Utility Manual*
- *VMS Network Control Program Manual*
- *VMS Networking Manual*
- *VMS Release Notes*
- *VMS Show Cluster Utility Manual*
- *VMS System Generation Utility Manual*
- *VMS System Manager's Manual*
- *VMS System Messages and Recovery Procedures Reference Manual*
- *VMS SYSMAN Utility Manual*
- *VMS User's Manual*
- *VMS VAXcluster Manual*

COURSE CONVENTIONS

Table 1 describes the conventions used in the listings and command tables of the 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, you would 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. For example, 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 of the data that the system would display in response to the particular command is shown, or that not all the data a user would enter is shown.
	`$ 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` Braces 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 (').

Managing Disks

INTRODUCTION

This chapter presents the basic concepts of disk management. Among the topics covered are:

- The conventions used for VMS device names
- The conventions used for VMS device names in a VAXcluster system
- How to modify characteristics of disk files
- How to use disk quotas to control disk space allocation

OBJECTIVES

To describe the tasks and responsibilities for maintaining disks, a system and network manager should be able to:

- Identify devices by name in a VAXcluster system
- Mount devices in a VAXcluster system
- Use VMS SHOW command to obtain information on system disk devices
- Create a volume set
- Modify several important file characteristics
- Use disk quotas to control the allocation of disk space to users

RESOURCES

- *VMS DCL Dictionary*
- *VMS System Manager's Manual*
- *Guide to Setting Up a VMS System*
- *Guide to Maintaining a VMS System*

TOPICS

- Review of VMS device names
- Device names in a VAXcluster system
- Using SHOW DEVICE to obtain information about disk volumes
- Creating volume sets
- Setting file characteristics
- Using disk quotas to manage disk space usage

REVIEW OF VMS DEVICE NAMES

Every device has a unique name in the format: **ddcu**

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

The device code specifies the device type.

The hardware controller number:

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

The unit number:

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

Table 1–1 lists some common two-letter device codes.

Table 1–1 Some Common Device Codes

Code	Device
CS	Console storage device
DU	RA80 or RA81 disk drive
MU	TK70 tape drive
LT	Terminal connected by means of a terminal server

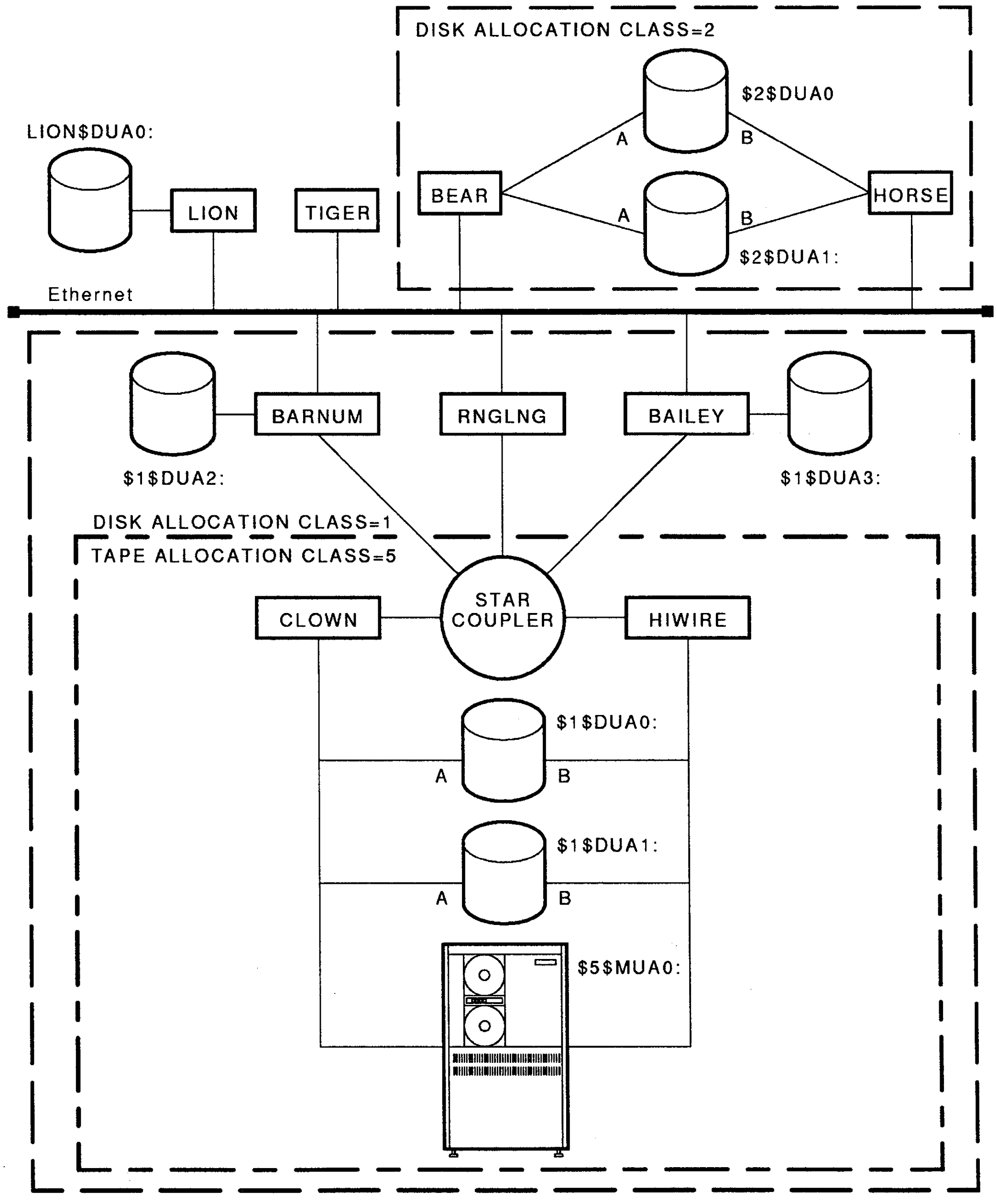
For example:

- **CSA1** = Console Storage, Controller A, Device 1
- **LTA251** = Local Area Transport, Controller A, Device 251

DEVICE NAMES IN A VAXcluster SYSTEM

Figure 1–1 shows a sample cluster with allocation classes assigned to nodes. It also shows the device name of each disk and tape drive.

Figure 1–1 Disk and Tape Device Names in a VAXcluster System Using Allocation Classes



MSCP Server

The mass storage control protocol (MSCP) server allows disks connected locally to a VAX processor or to an intelligent, MSCP compliant controller to be shared cluster-wide.

- These disks include:
  - Disks local to CI members
  - Disks on boot servers, and disk servers anywhere in the cluster
  - Disks on disk servers anywhere in the cluster, including satellites
  - HSC disks in mixed-interconnect clusters (when the MSCP server is running on one of the CI nodes)
  - Integrated storage element (ISE) disks connected to a DSSI (Digital standard storage interconnect) in a mixed-interconnect cluster
- The MSCP server decodes and services MSCP I/O requests sent by the disk class driver on remote cluster nodes.
  - Once a device is MSCP served, any processor in the cluster can mount the device and access it as if it were a local device.

Tape MSCP Server

The tape MSCP server peforms a similar function for tape drives.

- Once a tape device is MSCP served, any processor in the cluster can mount and access it as if it were a local device.
- Like a local tape drive, an MSCP served tape drive can be used by only one process at a time.

Device Name Formats

Two formats are used in naming devices:

- **node$device**
  - Used for devices that are directly connected to only one node.
  - Zero is the default value for the ALLOCLASS SYSGEN parameter, forcing this format.
- **$allocation-class$device**
  - Allocation class is a parameter set on a node that serves disks (an HSC or ISE controller or a VAX node running the MSCP server).
  - Zero is the default.
  - Setting the ALLOCLASS parameter to non-zero (1-255) enables this format for device names.
  - Used for dual-pathed (including dual-ported and dual-hosted) devices, to provide a single name for the device.
  - For any disk or tape device, all nodes that serve it to the cluster must have the same allocation class.
- No two devices in a cluster can have the same name.

  For example, if both BARNUM and BAILEY have allocation class 1:

  - You may not connect a drive named DUA0 to each node because there would then be two devices named 1DUA0.
  - To avoid this problem, change the unit number of one of the drives.

When a device is on a local node:

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

Use **node$device** to refer to devices on other nodes when specifying:

- A terminal on another node to OPCOM
- A printer on another node to the job controller

MOUNTING VOLUMES

To mount disk volumes in a common-environment VAXcluster system for the highest availability:

- Make local devices available cluster-wide through the MSCP server.
- Mount HSC based disks on all CI members using the MOUNT/SYSTEM command.
- Mount DSSI based ISEs on all DSSI members using the MOUNT/SYSTEM command.
- Mount MSCP served HSC, DSSI, and local disks on any or all nodes.
  - The command MOUNT/CLUSTER mounts a disk volume on all nodes currently in the cluster.
  - A node that joins the cluster later must explicitly mount the volume.
- So that all volumes remain mounted cluster-wide, the startup command procedure should:
  - Mount all local MSCP served disks with the MOUNT/CLUSTER command
  - Mount all remote disks with the MOUNT or MOUNT/CLUSTER command

Proper Dismount of Disks on Shutdown

Building a VAXcluster system should include setting up SYS$MANAGER:SYSHUTDWN.COM.

When you shut down a system:

- DISMOUNT/CLUSTER each MSCP served disk on the system that is not dual-ported.
  - Disks that are not dismounted undergo mount verification on other systems that have them mounted.
  - All processes with outstanding I/O to these disks hang.
  - If mount verification times out, the other systems must dismount and remount the disks.
- Dismount disks in the site-specific shutdown procedure SYS$MANAGER:SYSHUTDWN.COM
  - For example, on HORSE:

    ```
    $ DISMOUNT/CLUSTER/ABORT $2$DUA0:
    ```

  - /ABORT forces a disk to be dismounted even if it has open user files. This does not work if the open files are paging or swapping files, or if the disk is a system disk.

Rebuilding Incorrectly Dismounted Disks

Free space and storage allocation inconsistencies caused by incorrect dismounting of the system disk (for example, if there is a system failure) are fixed by rebuilding the disk.

- Rebuilding takes place automatically unless you use the MOUNT/NOREBUILD command to mount the disk.
- You should use MOUNT/NOREBUILD in the startup command procedure to mount each disk in a VAXcluster system.
  - Otherwise, processes using the disk hang until the rebuild completes.
- If the system disk is rebuilt at startup time, and the system disk is also being served by the MSCP server, the cluster can hang indefinitely.
  - To prevent this situation, on any system that serves a system disk or that boots from a served system disk you must use AUTOGEN to set the SYSGEN parameter ACP_REBLDSYSD to 0.
  - ACP_REBLDSYSD defaults to 0 for satellites; you must set it for boot servers.
- Rebuild disks at a more convenient time (such as in a batch job at an off-hour).

```
$ SET VOLUME/REBUILD SYS$SYSDEVICE
$ SET VOLUME/REBUILD $1$DUA1:
    .
    .
    .
```

  - If the disk does not need to be rebuilt, this command has no effect.
- There is no risk to data integrity by not rebuilding a disk at startup time. The worst consequence is that some free space is not available until the disk is rebuilt.

USING SHOW DEVICE TO OBTAIN INFORMATION ABOUT DISK VOLUMES

- SHOW DEVICE
  - Lists devices on the system
- SHOW DEVICE/FULL
  - Shows the complete status of a device
  - Useful for determining the configuration of disks in a cluster
- SHOW DEVICE/FILES
  - Lists the files that are currently open
  - This command lists files opened only on this node
- SHOW DEVICE D
  - To see a list of only disk devices:

```
$ SHOW DEVICE D

Device                   Device            Error    Volume          Free  Trans Mnt
 Name                    Status            Count     Label         Blocks Count Cnt
LION$DUA0:               Mounted               0  (remote mount)                  1
$1$DUA0:      (CLOWN)    Mounted               0  THREERING       195039   417   7
$1$DUA1:      (HIWIRE)   Mounted               0  TIGHTROPE       223851     1   7
$1$DUA2:      (BARNUM)   Mounted               0  FLYING          261060     1   7
$1$DUA3:      (BAILEY)   Mounted               0  TRAPEZE         174615     6   7
$2$DUA0:      (BEAR)     Mounted               0  ELEPHANT        195039   417   7
$2$DUA1:      (BEAR)     Mounted               0  BALLERINA       223851     1   7
```

Example 1–1 SHOW DEVICE/FULL for a Locally Connected Disk

```
$ SHOW DEVICE/FULL $1$DUA2

Disk $1$DUA2:(BARNUM), device type RA81,❶ is online, mounted,
❷file-oriented device,
    shareable, served to cluster via MSCP Server, error logging is enabled.

    Error count                    0❼  Operations completed            5989
    Owner process                 ""   Owner UIC                     [1,1]❻
    Owner process ID        00000000   Dev Prot   S:RWED,O:RWED,G:RWED,W:RWED❻
    Reference count                1   Default buffer size              512
    Total blocks             891072❹  Sectors per track                 51
    Total cylinders             1248   Tracks per cylinder                8
    Allocation class               1

    Volume label           "FLYING"   Relative volume number             0
    Cluster size                   3   Transaction count                 93
    Free blocks                8069❺  Maximum files allowed         222768
    Extend quantity                5   Mount count                        7
    Mount status              System   Cache name         "_$1$DUA0:XQPCACHE"
    Extent cache size             64   Maximum blocks in extent cache  2088
    File ID cache size            64   Blocks currently in extent cache   0
    Quota cache size              30   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 RNGLNG, BAILEY, LION, HORSE, BEAR, TIGER.❸
```

This example answers questions including the following:

*allocate 3 blocks at a time

Done only at initialization time.

❶ What type of disk is it?

❷ Is it mounted on the local system?

❸ Is it mounted on any other system in the cluster?

❹ How big is it? (in 512-byte blocks)

❺ How many blocks are free?

❻ What is the owner UIC and volume protection?

❼ Has it generated any hardware errors since the system was started up?

Example 1–2 SHOW DEVICE/FULL for a Remote Disk

```
$ SHOW DEVICE/FULL $1$DUA2

Disk $1$DUA2:(BARNUM), device type RA81, is online, mounted, file-oriented
    device, shareable, available to cluster, error logging is enabled.

    Error count                    0    Operations completed              5989
    Owner process                 ""    Owner UIC                        [1,1]
    Owner process ID        00000000    Dev Prot    S:RWED,O:RWED,G:RWED,W:RWED
    Reference count                1    Default buffer size                512
    Total blocks              891072    Sectors per track                   51
    Total cylinders             1248    Tracks per cylinder                  8
    Host name               "BARNUM"❶   Host type, available     VAX 8810, yes❷
    Allocation class               1

    Volume label            "FLYING"    Relative volume number               0
    Cluster size                   3    Transaction count                   93
    Free blocks                 8069    Maximum files allowed           222768
    Extend quantity                5    Mount count                          7
    Mount status              System    Cache name         "_$1$DUA2:XQPCACHE"
    Extent cache size             64    Maximum blocks in extent cache    2088
    File ID cache size            64    Blocks currently in extent cache     0
    Quota cache size              30    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 BARNUM, RNGLNG, LION, HORSE, BEAR, TIGER.
```

Additional information you might want to know about a remote disk:

❶ Which cluster node is the disk connected to?

❷ What type of node is it connected to?

Example 1–3 SHOW DEVICE/FULL for Dual-Ported HSC Disk

```
$ SHOW DEVICE/FULL $1$DUA1

Disk $1$DUA1: (HIWIRE), device type RA82, is online, mounted, file-oriented
    device, shareable, available to cluster, error logging is enabled.

    Error count                    2    Operations completed         2352587

    Owner process                 ""    Owner UIC                      [1,1]
    Owner process ID        00000000    Dev Prot     S:RWED,O:RWED,G:RWED,W:RWED
    Reference count               76    Default buffer size              512
    Total blocks             1216665    Sectors per track                 51
    Total cylinders             1248    Tracks per cylinder               14
    Host name               "HIWIRE"❶   Host type, available      HSC50, yes❷
    Alternate host name      "CLOWN"❶   Host type, available      HSC70, yes❷
    Allocation class               1

    Volume label         "TIGHTROPE"    Relative volume number             0
    Cluster size                   1    Transaction count                223
    Free blocks               121857    Maximum files allowed         222768
    Extend quantity                5    Mount count                        7
    Mount status              System    Cache name       "_$1$DUA1:XQPCACHE"
    Extent cache size             64    Maximum blocks in extent cache   806
    File ID cache size            64    Blocks currently in extent cache  70
    Quota cache size               0    Maximum buffers in FCP cache     350

  Volume status:  subject to mount verification, file high-water marking, write-
      through caching enabled.
  Volume is also mounted on BAILEY, LION, HORSE, RNGLNG, BEAR, TIGER.
```

Information about a disk connected between two HSC nodes:

❶ What controllers is it connected to?

❷ What types of controllers is it connected to?

SHOW DEVICE/FILES

Shows which files on a particular device are open by processes on the local system.

Example 1–4 SHOW DEVICE/FILES Output

```
$ SHOW DEVICE/FILE $1$DUA1:
Files accessed on device _$1$DUA1: on  9-MAY-1989 11:47:44.50

Process name      PID      File name
                00000000  [000000]INDEXF.SYS;1
                00000000  [000000]QUOTA.SYS;1
T O M           20202580  [JAGGER.MAIL]MAIL.MAI;1
Bette           2020267C  [FINNERN.OLTP]OBJECTIVES.TJL;1
Ed Bernstein    2020233A  [BERNSTEIN]MYEVEPLUS.TPU$SECTION;44
Ed Bernstein    2020233A  [BERNSTEIN.CLUSTER]LL1.TJL;1
Mike Beeler     2020275B  [BEELER.CLUSTER.LL]TEST_IT.DAT;1
Mike Beeler     2020275B  [BEELER.CLUSTER.LL]D.COM;1
Mike Beeler     2020275B  [BEELER.CLUSTER.LL]T.LIS;1
```

Reasons you might want this information:

- You cannot dismount a disk volume because it has open files on it.
  — The output shows you which files are open and by which process.
- A user cannot open a shared data file because another user already has it open.
  — Look for that file in the output and note which user's process has it open.

You can also use SHOW DEVICE using SYSMAN to list all of the open files in a cluster.

Example 1–5 SHOW DEVICE Using SYSMAN Output

```
$ SET DEFAULT SYS$SYSTEM
$ SET PROCESS/PRIVILEGE=SYSPRV
$ RUN SYSMAN
SYSMAN> SET ENVIRONMENT/CLUSTER
%SYSMAN-I-ENV, current command environment:
        Clusterwide on local cluster
        Username MATTHEWS     will be used on nonlocal nodes

SYSMAN> DO SHOW DEVICE/FILES $2$DUA0
%SYSMAN-I-OUTPUT, command execution on node BARNUM

Files accessed on device $2$DUA0: (BEAR) on  4-SEP-1991 17:38:10.89

Process name      PID     File name
                00000000  [000000]INDEXF.SYS;1
%SYSMAN-I-OUTPUT, command execution on node RNGLNG

Files accessed on device $2$DUA0: (BEAR) on  4-SEP-1991 17:37:37.71

Process name      PID     File name
                00000000  [000000]INDEXF.SYS;1
Livvy  0        23200095  [BUNNELL]DECW$SM.LOG;40
                00000000  [MANDRA.SYSEXE]PAGEFILE1.SYS;1
%SYSMAN-I-OUTPUT, command execution on node BEAR

Files accessed on device $2$DUA0: (BEAR) on  4-SEP-1991 17:37:58.38

Process name      PID     File name
                00000000  [000000]INDEXF.SYS;1
DECW$SESSION    23600058  [ROUNDS]DECW$SM.LOG;222

    .
    .
    .
```

CREATING VOLUME SETS

If the files or user directories become too large to fit on any one volume, you can create a *volume set*.

- Two or more disk volumes
- Bound together with the MOUNT/BIND command

The VMS operating system:

- Treats a volume set as one large volume
- Stores files on any volume in the set that has free space
- Attempts to use space evenly over all volumes in a set

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
```

Example 1–6 illustrates how to create a public volume set (USER_SET) starting with an existing volume (USER1) as the root volume.

Example 1–6 illustrates how to create a volume set from an existing volume.

Example 1–6 Creating a Volume Set from an Existing Volume

```
1 $ ALLOCATE DRA2 DEV1
%DCL-I-ALLOC, _DRA2: allocated
$ ALLOCATE DRA3 DEV2
%DCL-I-ALLOC, _DRA3: allocated
2 $ INITIALIZE/SYSTEM DEV2 USER2
3 $ MOUNT/SYSTEM/BIND=USER_SET DEV1,DEV2 USER1,USER2
%MOUNT-I-MOUNTED, USER1        mounted on _DRA2:
%MOUNT-I-MOUNTED, USER2        mounted on _DRA3:
4 $ 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]"
$
5 $ 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.
6 $ DISMOUNT DEV1
$ SHOW LOGICAL/SYSTEM D*

(LNM$SYSTEM_TABLE)

   "DBG$INPUT" = "SYS$INPUT:"
   "DBG$OUTPUT" = "SYS$OUTPUT:"
7  "DDP$DIS" = "SYS$MANAGER:DDP.DIS"
   "DISK$VAXVMSRL052" = "DUA0:"
   "DTR$LIBRARY" = "SYS$SYSROOT:[DTR]"
$
```

Notes on Example 1–6:

❶ 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 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 The **MOUNT** command string binds the volumes into the disk volume set, USER_SET. The **/SYSTEM** qualifier makes the set public. You must have SYSNAM privilege to use it. The root volume (USER1) contains the directory structure for the entire volume set.

❹ 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. For example, 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 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 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.

USING DISK QUOTAS TO MANAGE DISK SPACE USAGE

The UAF restricts use of many system resources, but there is no value in the UAF record that restricts use of disk space.

Disk space restriction is handled through **disk quotas**. Quotas are based on UICs, not individual user names.

Disk quotas are managed through use of the SYSMAN utility **DISKQUOTA** command subset.

Disk quotas are enabled on a volume-by-volume basis; the default is no disk quotas enabled.

Quota files:

- One file per enabled volume: [000000]QUOTA.SYS
- Contains **quota entries,** one per UIC
- Created and manipulated by the SYSMAN utility

Establishing Quotas on a Volume

A quota file must be created in the volume's MFD (directory [000000]). Exact steps for properly creating the quota file depend on whether:

- The volume has just been created (no user files exist)
- The volume has been in use for a while (user files already exist)

One entry must be created for each UIC allowed to use the volume. Each quota entry contains the following fields:

- UIC
- Usage
- Permanent quota
- Overdraft

NOTE

Quotas should not be enabled on the system disk.

Table 1–2 illustrates establishing quotas on a new volume.

Table 1–2 Establishing Quotas on a New Volume Called DISK$DATA

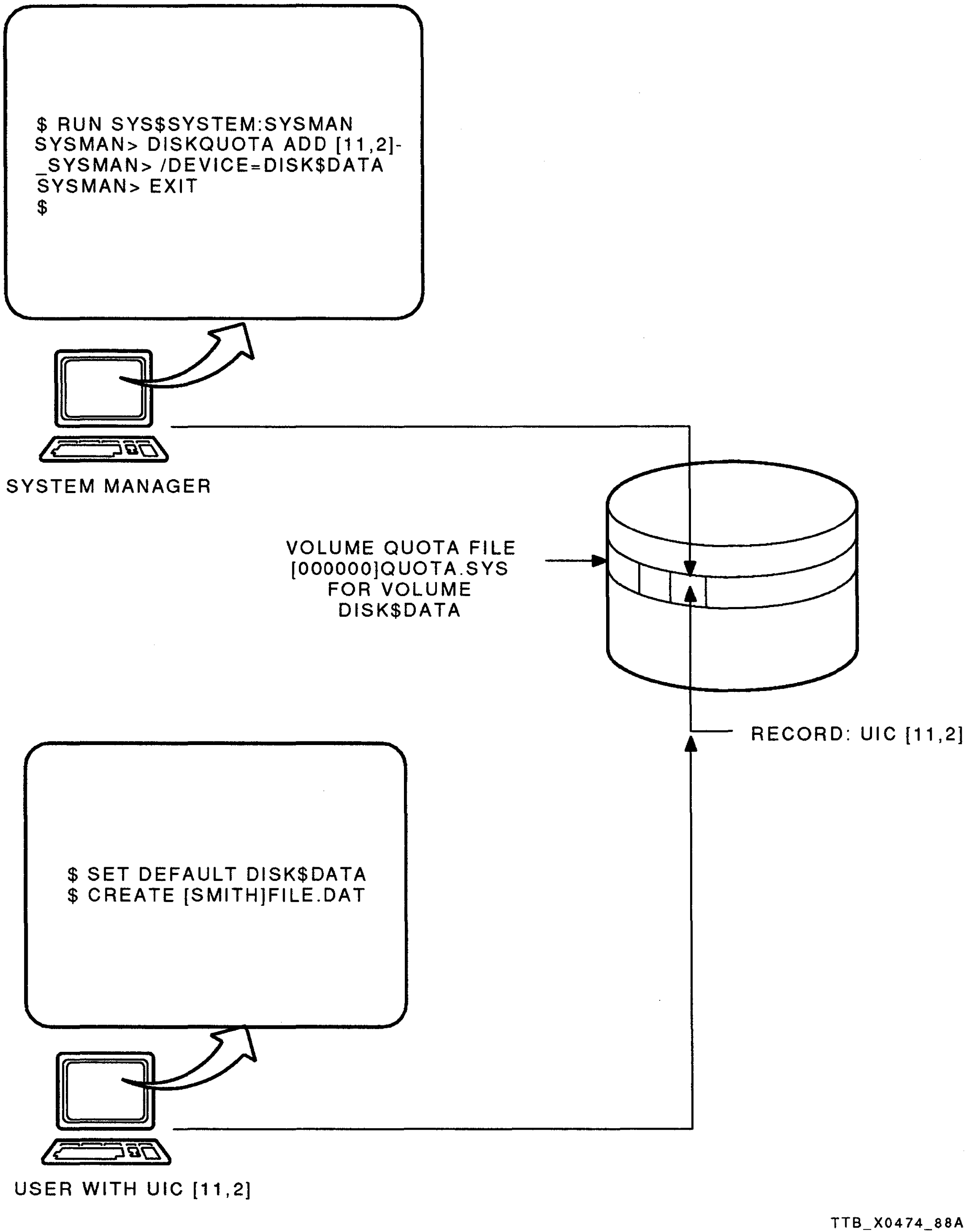
Steps	Commands	Comments
1	Log in as SYSTEM	You can alternatively give your current process OPER privilege to use the SYSMAN utility.
2	`$ RUN SYS$SYSTEM:SYSMAN`	Invokes the SYSMAN utility.
3	`SYSMAN> DISKQUOTA CREATE -` `_SYSMAN> /DEVICE=DISK$DATA`	The file DISK$DATA:[000000]QUOTA.SYS is created and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

Table 1–3 shows the fields in a quota file record.

Table 1–3 Fields in a Quota File Record

Field	Meaning	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 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 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 the VMS system 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 the VMS system refuses to create new files or extend existing files. Therefore, the permanent quota plus the overdraft define the total number of blocks available to a user on a volume.	`/OVERDRAFT`

Figure 1–2 Adding a Quota Record to a Volume Quota File



Example 1–7 shows output from the SYSMAN DISKQUOTA SHOW command.

Example 1–7 List of Volume Quota File Records

```
$ 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
[11,2]                    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
$
```

Table 1–4 lists commands for displaying the contents of a volume quota file.

Table 1–4 Displaying the Contents of a Volume Quota File

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

NOTE

The disk volume usage recorded by the VMS 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 usually different from the disk usage displayed by the DCL command DIRECTORY/SIZE=ALLOCATED.

Table 1–5 illustrates SYSMAN DISKQUOTA commands.

Table 1–5 Managing Individual Records in the Volume Quota File

Operation [1]	SYSMAN Command Format
Adds a new entry, specifying values different from the default entry ([0,0])	`SYSMAN>DISKQUOTA ADD uic -` `_SYSMAN> [/PERMQUOTA=blks1] [/OVERDRAFT=blks2]`
Modifies an existing entry	`SYSMAN>DISKQUOTA MODIFY uic -` `_SYSMAN> [PERMQUOTA=blks1] [/OVERDRAFT=blks2]`
Modifies the entry for [0,0], used to supply default values for permanent quota and overdraft. ([0,0] should never own any files.)	`SYSMAN>DISKQUOTA MODIFY [0,0] -` `_SYSMAN> [/PERMQUOTA=blks1] [/OVERDRAFT=blks2]`
Modifies all entries for UICs in a particular group	`SYSMAN>DISKQUOTA MODIFY [group-number,*] -` `_SYSMAN> [/PERMQUOTA=blks1] [/OVERDRAFT=blks2]`
Modifies all entries, including the default entry, [0,0]	`SYSMAN>DISKQUOTA MODIFY [*,*] -` `_SYSMAN> [/PERMQUOTA=blks1] [/OVERDRAFT=blks2]`
Removes an existing entry	`SYSMAN>DISKQUOTA REMOVE uic`

[1] 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

Table 1–6 illustrates establishing quotas on an existing volume whose logical name is DISK$USER.

Table 1–6 Establishing Quotas on an Existing Volume Called DISK$USER

Step	Commands	Comments
1	Notify users that DISK$USER will be unavailable	The REPLY command and the Mail utility are two possible methods.
2	`$ RUN SYS$SYSTEM:SYSMAN`	Invokes the SYSMAN utility. Make sure your current process has OPER privilege (to use the SYSMAN utility) and SYSPRV privilege (to issue DISKQUOTA commands).
3	`SYSMAN> DISKQUOTA -` `_SYSMAN> CREATE /DEVICE=DISK$USER`	Creates a quota file on the DISK$USER volume (DISK$USER:[000000]QUOTA.SYS) and automatically enables quotas on that volume.

4	`SYSMAN> DISKQUOTA MODIFY [0,0] -`
	`_SYSMAN> /PERMQUOTA=10000 -`
	`_SYSMAN> /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.

5	`SYSMAN> DISKQUOTA REBUILD`	Updates the newly created quota file to add existing UICs that own files on the DISK$USER volume. Note that the **/DEVICE=DISK$USER** qualifier need not be specified in this case.
6	`SYSMAN> EXIT`	Exits from the SYSMAN utility.
7	Notify users that DISK$USER is available for use	See step 1.

Disabling and Enabling Quotas on a Volume

To disable quotas, use the SYSMAN **DISKQUOTA DISABLE** command.

Use the SYSMAN **DISKQUOTA REBUILD** command to properly assess user space on a volume if:

- The volume previously had a quota file created
- Quotas were once enabled on the volume
- Quotas were later disabled

An automatic **DISKQUOTA REBUILD** is performed when a volume is mounted after being improperly dismounted. This is a typical situation after a system failure.

Always Enabled on Reboot

SETTING FILE CHARACTERISTICS

Another way to manage the use of disk space is to use the SET FILE command, which modifies the characteristics of files.

- Format:

```
$ SET FILE/qualifier(s)  file-spec
```

- The *file-spec* specifies one or more files to be modified.
  - — If you specify more than one file, separate them with commas. (Wildcard characters are allowed.)
- Some of the file characteristics you can modify are listed in Table 1–7.

Table 1–7 Some Qualifiers for the SET FILE Command

Qualifier	Description
Specifying File Characteristics	
/EXPIRATION_DATE=date /NOEXPIRATION_DATE	Assigns an expiration date to the specified files. Later, you can use DELETE/EXPIRED to delete files whose expiration date has passed.
/VERSION_LIMIT*[=n]*	Specifies the maximum number of versions allowed for the specified file. If the version limit is exceeded, the earliest version of the file is deleted from the directory without notification to the user.
/OWNER_UIC*[=uic]*	Specifies an owner UIC for the file. The default is the UIC of the current process. Useful if you create a file in a user's directory and you want the user to own the file. VMS 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.
/Prot=(...)	
Modifying the Action of the SET FILE Command	
/BY_OWNER*[=uic]*	Selects only those files whose owner UIC matches the specified UIC. The default is the UIC of the current process.
/CONFIRM /NOCONFIRM	Controls whether a request is issued before each SET FILE operation to confirm that the operation should be performed on that file. The default is /NOCONFIRM.
/EXCLUDE=(file-spec*[,...]*)	Excludes the specified file from the SET FILE operation. (Wildcard characters are allowed.)
/LOG	Displays the file specification of each file modified as the command executes.

Example 1–8 illustrates the use of some of these qualifiers.

/ENTER = alias-name

Example 1–8 SET FILE Command

```
❶ $ DIR/FULL TEST.COM

  Directory WHYSO$DUA0:[ROUNDS]

  TEST.COM;1                    File ID:  (1620,1,0)
❷ Size:             1/3         Owner:    [ROUNDS]
  Created:   9-JUN-1990 14:25:03.66
  Revised:  22-MAY-1991 16:00:20.22 (3)
❸ Expires:   <None specified>
  Backup:    <No backup recorded>
  File organization:  Sequential
  File attributes:    Allocation: 3, Extend: 0, Global buffer count: 0
❹                     No version limit
  Record format:      Variable length, maximum 12 bytes
  Record attributes:  Carriage return carriage control
  RMS attributes:     None
  Journaling enabled: None
  File protection:    System:RWED, Owner:RWED, Group:RE, World:
  Access Cntrl List:  None

  Total of 1 file, 1/3 blocks.
  $
❺ $ SET FILE /OWNER=MARSH /EXPIRATION_DATE=0 /VERSION_LIMIT=3-
  $_   /LOG TEST.COM
❻ %SET-I-MODIFIED, WHYSO$DUA0:[ROUNDS]TEST.COM;1 modified
  $
❼ $ DIR/FULL TEST.COM

  Directory WHYSO$DUA0:[ROUNDS]

  TEST.COM;1                    File ID:  (1620,1,0)
❽ Size:             1/3         Owner:    [MARSH]
  Created:   9-JUN-1990 14:25:03.66
  Revised:  25-MAY-1991 11:21:37.92 (4)
❾ Expires:  25-MAY-1991 00:00:00.00
  Backup:    <No backup recorded>
  File organization:  Sequential
  File attributes:    Allocation: 3, Extend: 0, Global buffer count: 0
❿                     Version limit: 3
  Record format:      Variable length, maximum 12 bytes
  Record attributes:  Carriage return carriage control
  RMS attributes:     None
  Journaling enabled: None
  File protection:    System:RWED, Owner:RWED, Group:RE, World:
  Access Cntrl List:  None

  Total of 1 file, 1/3 blocks.
  $
```

(Notes on Example 1–8 are shown on the next page.)

Notes on Example 1–8:

1. Display full directory information about TEST.COM.
2. The owner UIC of the file is [ROUNDS].
3. The file has no expiration date.
4. The file has no version limit.
5. Change several characteristics of the file and request that the name of the file be displayed when the operation takes place.
6. The log message.
7. Redisplay the directory information.
8. The owner UIC of the file is now [MARSH].
9. The file now has an expiration date that defaults to the current date since 0 was specified as the date.
10. The file now has a version limit of 3.

SUMMARY

Disk and tape drives are devices that record and read data on magnetic media or optical disks.

- Every device has a unique name in the format of **ddcu**.
- There are two formats used in naming devices in a VAXcluster system:
  - node$device
  - $allocation-class$device
- The MOUNT/CLUSTER command mounts a volume on every node that is currently a member of the cluster, and the DISMOUNT/CLUSTER command dismounts a volume on every cluster node that has it mounted.
- To suppress a volume rebuild, which can cause processes (or the entire cluster in the case of a system disk) to hang, use the MOUNT/NOREBUILD command.
  - The disk should be rebuilt later with the SET VOLUME/REBUILD command.
- The SHOW DEVICE command is used to obtain information about disk volumes.
- The SET FILE command can be used to modify file characteristics including expiration date, version limit, and owner UIC.
- The SYSMAN utility is used to set quotas on the use of disk space. Use it to:
  - Establish disk quotas on a volume, whether it is new or already in use
  - Add user entries to the quota file on a disk volume

Using Logical Name Tables

INTRODUCTION

Logical names provide the VMS user with a convenient way of referring to programs, command procedures, disk locations, and other objects without having to know their physical location. Users can define their own logical names, and the system manager can also define logical names to make them available to users.

The VMS operating system stores all system-defined and user-defined logical names in tables, called *logical name tables*.

This chapter describes the features related to logical name tables.

OBJECTIVES

To further customize the VMS working environment, a system and network manager needs to be able to:

- Determine the equivalence of a logical name
- Display the contents of logical name tables
- Use and identify some system-created logical names
- Define logical names in the system table

RESOURCES

- *Guide to Using VMS Command Procedures*
- *VMS DCL Dictionary*
- *Guide to Using VMS*
- *VMS User's Manual*
- *VMS DCL Concepts Manual*

TOPICS

- Logical name tables
- Logical name translation
- Search lists
- System-created logical names
- Defining names in the system table
- Duration of logical names

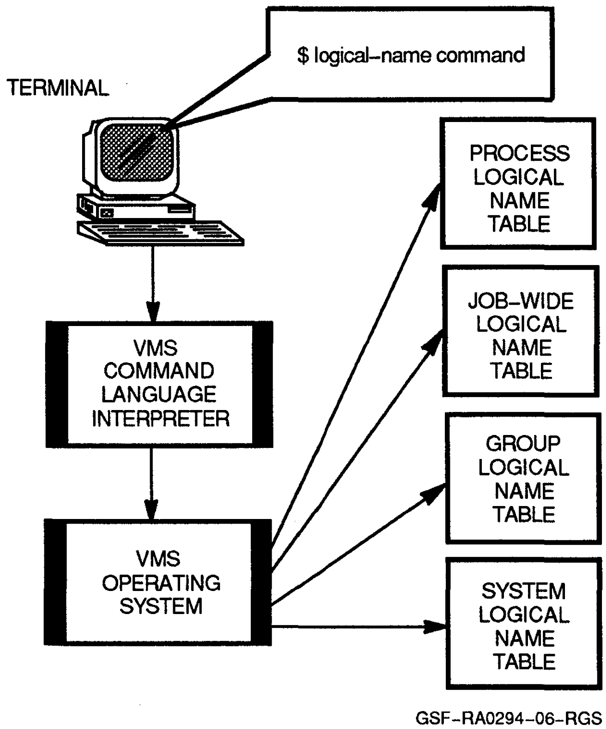
LOGICAL NAME TABLES

The system stores logical names and their equivalence strings in four default logical name tables and possibly additional user-defined tables:

- **Process logical names**
  - Used only by your process
  - Stored in process logical name table
- **Job logical names**
  - Shared by your process and all of its subprocesses
  - Stored in job logical name table
  - DCL commands that use this table include the qualifier /JOB
- **Group logical names**
  - Shared by all processes in a UIC group
  - Stored in group logical name table
  - GRPNAM privilege is needed to add logical names to this table
  - DCL commands that use this table include the qualifier /GROUP
- **System logical names**
  - Shared by all processes on the system
  - Stored in system logical name table
  - SYSNAM privilege is needed to add logical names to this table
  - DCL commands that use this table include the qualifier /SYSTEM
- **Other logical names**
  - Used for special applications
  - For example, the table DECW$LOGICAL_NAMES holds logical names used by DECwindows software
  - DCL commands that use these tables include the qualifier /TABLE=

Figure 2–1 illustrates the relationship between your terminal, the operating system, and the logical name tables associated with your process.

Figure 2–1 Relationship Between Your Terminal, the Operating System, and Logical Name Tables



The SHOW LOGICAL command, shown in Example 2–1, displays the contents of all logical name tables to which you have access.

Example 2–1 Displaying the Contents of Logical Name Tables

```
$ SHOW LOGICAL

(LNM$PROCESS_TABLE)

  "SYS$COMMAND" = "_TIDY$LAT2:"
  "SYS$DISK" = "TIDY$DJA0:"
  "SYS$ERROR" = "_TIDY$LTA2:"
  "SYS$INPUT" = "_TIDY$LTA2:"
  "SYS$OUTPUT" [super] = "_TIDY$LTA2:"
  "SYS$OUTPUT" [exec] = "_TIDY$LTA2:"
  "TT" = "LTA2:"

(LNM$JOB_803DD220)

  "SYS$LOGIN" = "TIDY$DJA0:[HENDRICKS]"
  "SYS$LOGIN_DEVICE" = "TIDY$DJA0:"
  "SYS$SCRATCH" = "TIDY$DJA0:[HENDRICKS]"

(LNM$GROUP_000011)

(LNM$SYSTEM_TABLE)

  "ACP$BADBLOCK_MBX" = "MBA4:"
  "DBG$INPUT" = "SYS$INPUT:"
  "DBG$OUTPUT" = "SYS$OUTPUT:"
    .
    .
    .
  "SYS$STARTUP" = "SYS$SYSROOT:[SYS$STARTUP]"
 = "SYS$MANAGER"
  "SYS$SYLOGIN" = "SYS$MANAGER:SYLOGIN.COM"
  "SYS$SYSDEVICE" = "TIDY$DJA0:"
  "SYS$SYSDISK" = "SYS$SYSROOT:"
  "SYS$SYSROOT" = "TIDY$DJA0:[SYS0.]"
 = "SYS$COMMON:"
  "SYS$SYSTEM" = "SYS$SYSROOT:[SYSEXE]"
  "SYS$TEST" = "SYS$SYSROOT:[SYSTEST]"
  "SYS$TOPSYS" = "SYS0"
  "SYS$UPDATE" = "SYS$SYSROOT:[SYSUPD]"
  "SYS$WELCOME" = "@SYS$MANAGER:WELCOME.TXT"
  "TPU$SECTION" = "EVE$SECTION"

(DECW$LOGICAL_NAMES)

  "CDA$LIBRARY" = "SYS$COMMON:[CDA$LIBRARY]"
  "DECW$BOOK" = "SYS$COMMON:[DECW$BOOK]"
  "DECW$EXAMPLES" = "SYS$COMMON:[SYSHLP.EXAMPLES.DECW]"
    .
    .
    .
```

Example 2–2 shows how to display the contents of the process, job, group, system, and DECwindows logical name tables using the /PROCESS, /JOB, /GROUP, /SYSTEM, and /TABLE qualifiers.

Example 2–2 Displaying the Contents of Logical Name Tables

```
$ SHOW LOGICAL/PROCESS

(LNM$PROCESS_TABLE)

  "SYS$COMMAND" = "_TIDY$LTA12:"
  "SYS$DISK" [super] = "VMS$COM:"
  "SYS$DISK" [exec] = "VMS$COM:"
  "SYS$ERROR" = "_TIDY$LTA12:"
  "SYS$INPUT" = "_TIDY$LTA12:"
  "SYS$OUTPUT" [super] = "_TIDY$LTA12:"
  "SYS$OUTPUT" [exec] = "_TIDY$LTA12:"

$ SHOW LOGICAL/JOB

(LNM$JOB_803E4E40)

  "SYS$LOGIN" = "DISK:[SMITH]"
  "SYS$LOGIN_DEVICE" = "DISK:"
  "SYS$SCRATCH" = "DISK:[SMITH]"

$ SHOW LOGICAL/GROUP

(LNM$GROUP_000011)

  "GROUP11_DISK" = "DJAO:"

$ SHOW LOGICAL/SYSTEM

(LNM$SYSTEM_TABLE)

  "DBG$INPUT" = "SYS$INPUT:"
  "DBG$OUTPUT" = "SYS$OUTPUT:"
   .
   .
   .

$ SHOW LOGICAL/TABLE=DECW$LOGICAL_NAMES

(DECW$LOGICAL_NAMES)

  "CDA$LIBRARY" = "SYS$COMMON:[CDA$LIBRARY]"
  "DECW$BOOK" = "SYS$COMMON:[DECW$BOOK]"
   .
   .
   .
```

LOGICAL NAME TRANSLATION

The system translates logical names automatically.

- By default, logical name tables are searched for the first occurrence of a logical name in the following order:
  1. Process logical name table
  2. Job-wide logical name table
  3. Group logical name table
  4. System logical name table
  5. Other logical name tables (for example, the DECwindows table)
- The leftmost portion of a file specification is translated to see if it is a logical name.

Determining the Equivalence of a Logical Name

- Two commands are available to determine the equivalence of a logical name.
- Command format:
  — `$ SHOW LOGICAL logical-name`

    Iteratively translates the logical name up to 10 levels

  — `$ SHOW TRANSLATION logical-name`

    Displays the **first** equivalence string it finds and stops (no iteration is performed)
- Translation is done:
  — Up to 10 times (recursively)

  — Until there are no more equivalence names to be translated

  — Until the leftmost component of the specification is not delimited by a colon, a space, a comma, or an end of line

  — Until equivalence name is a logical name that has the TERMINAL attribute. If a logical name has the TERMINAL attribute, the translation is "TERMINAL" (completed) after the first translation

  — If the logical name has the CONCEALED attribute, the translation normally displays the logical name for the device, rather than the physical name for the device

Example 2–3 shows both commands used to determine a logical name value.

Example 2–3 Determining the Value of a Logical Name

```
❶ $ ASSIGN DJAO: DISK1
❷ $ ASSIGN DISK1: MYNAME
❸ $ SHOW TRANSLATION MYNAME
    MYNAME = "DISK1:"  (LNM$PROCESS_TABLE)
❹ $ SHOW LOGICAL MYNAME
     "MYNAME" = "DISK1:" (LNM$PROCESS_TABLE)
  1  "DISK1" = "DJAO:" (LNM$PROCESS_TABLE)
```

Notes on Example 2–3

❶ Create a logical name, DISK1, that translates into the string "DJA0:".

❷ Create a second logical name, MYNAME, which translates to the string "DISK1".

❸ When you display the equivalence of a logical name using the SHOW TRANSLATION command, only one level of translation occurs.

❹ Note that the SHOW LOGICAL command translates the logical name iteratively.

Modifying Logical Name Translation

When creating a logical name, you can establish translation attributes that modify the interpretation of an equivalence name.

- Use the /TRANSLATION_ATTRIBUTES qualifier with the ASSIGN or DEFINE command.
- Two translation attributes are available:
  - — TERMINAL
  - — CONCEALED

Preventing Iterative Translation

The TERMINAL attribute prevents iterative translation of a logical name.

- The equivalence name is not examined to see if it is also a logical name.
- The translation is *terminal* (complete) after the first translation.

Concealing the True Identity of a Logical Name

The CONCEALED attribute causes the logical name of a device, rather than the physical name, to be displayed in system messages. This technique allows you to:

- Use a name that is more meaningful to users than the physical device name
- Hide physical device names from users, so that you can change a physical device without affecting users

Example 2–4 shows the use of the CONCEALED attribute.

Example 2–4 CONCEALED Attribute

```
$ ASSIGN/SYSTEM DUA0: USERDISK
$ DIRECTORY USERDISK:[ROUNDS]

Directory DUA0:[ROUNDS]

TEST.COM;5          TEST.DIR;1

Total of 2 files.
$
$ ASSIGN/TRANSLATION_ATTRIBUTES=CONCEALED/SYSTEM DUA0: USERDISK
%DCL-I-SUPERSEDE, previous value of USERDISK has been superseded
$ DIRECTORY USERDISK:[ROUNDS]

Directory USERDISK:[ROUNDS]

TEST.COM;5          TEST.DIR;1

Total of 2 files.
```

Once you have defined a concealed logical name for a device, you may want to specify it as a user's default device name, as shown in Example 2–5.

Example 2–5 Assigning a User to a Logical Default Device

```
$ RUN SYS$SYSTEM:AUTHORIZE
UAF> MODIFY ROUNDS /DEFAULT=USERDISK
UAF> SHOW ROUNDS

Username: ROUNDS                          Owner:  Kristin Rounds
Account:  GROUP22                         UIC:    [22,456] ([GROUP22,ROUNDS])
CLI:      DCL                             Tables:
Default:  USERDISK:[ROUNDS]
    .
    .
    .
```

SEARCH LISTS

A search list is a logical name that has more than one equivalence string. You can use a search list in any place that you can use a logical name.

Example 2–6 demonstrates the use of search lists.

Example 2–6 Search Lists

```
❶ $ DIRECTORY
  %DIRECT-W-NOFILES, no files found
  $
❷ $ CREATE MON.DAT,TUE,WED,THU,FRI
  This is MON CTRL/Z
  This is TUE CTRL/Z
  This is WED CTRL/Z
  This is THU CTRL/Z
  This is FRI CTRL/Z
  $
❸ $ DIRECTORY

  Directory TEST_DISK:[ROUNDS]

  FRI.DAT;1          MON.DAT;1          THU.DAT;1          TUE.DAT;1
  WED.DAT;1

  Total of 5 files.
  $
❹ $ DEFINE DAY MON.DAT,TUE.DAT,WED.DAT,THU.DAT,FRI.DAT
  $
❺ $ SHOW LOGICAL DAY
     "DAY" = "MON.DAT" (LNM$PROCESS_TABLE)
          = "TUE.DAT"
          = "WED.DAT"
          = "THU.DAT"
          = "FRI.DAT"
  $
❻ $ DIRECTORY DAY

  Directory TEST_DISK:[ROUNDS]

  MON.DAT;1          TUE.DAT;1          WED.DAT;1          THU.DAT;1
  FRI.DAT;1

  Total of 5 files.
  $
❼ $ TYPE DAY
  This is MON
  $
❽ $ DELETE/LOG MON.DAT;
  %DELETE-I-FILDEL, TEST_DISK:[ROUNDS]MON.DAT;1 deleted (3 blocks)
  $
❾ $ TYPE DAY
  This is TUE
```

(Notes on this example are shown on the next page)

Notes on Example 2-6:

❶ Check to see if there are any files in this directory.

❷ Create five files with one command.

The file type of the first file carries over to the other files because no file type was specified for them.

❸ Display the names of the files in the directory.
(Note that the names are shown in alphabetical order.)

❹ Create the logical name DAY, which is a search list because it has more than one equivalence name.

❺ Display the equivalences of the logical name DAY.
(Note that they are listed in the order in which they were specified in the DEFINE command.)

❻ Request directory information by specifying the logical name.
(Note that the file specifications are shown in the order of the search list.)

❼ Display the contents of the file that corresponds to the logical name DAY.
(Note that this file is the first element in the search list.)

❽ Delete MON.DAT, the first element in the search list.

❾ Now the logical name DAY is translated to TUE.DAT, and the contents of that file will be displayed.

Using Commas in Logical Name Assignments

- The presence of a comma generally indicates a search list.
- Sometimes you may want to use a comma as part of the actual equivalence string to save some typing in MAIL.

  You must enclose the equivalence string in quotation marks to indicate that the comma is a valid part of the string.
- Example 2–7 shows commas in logical name assignments.

Example 2–7 Commas in Logical Name Assignments

```
❶ $ DEFINE REVIEW BOOT,VASSILOS,APON
  $ MAIL

  MAIL> SEND
❷ To: REVIEW
  Subj: THIS IS A TEST
  Enter your message below. Press CTRL/Z when complete, or CTRL/C to quit:
  CTRL/Z

❸ MAIL> EXIT
  $
❹ $ DEFINE REVIEW "BOOT,VASSILOS,APON"
  %DCL-I-SUPERSEDE, previous value of REVIEW has been superseded
  $ MAIL

  MAIL> SEND
  To: REVIEW
  Subj: THIS IS ANOTHER TEST
  Enter your message below. Press CTRL/Z when complete, or CTRL/C to quit:
  CTRL/Z
❺ New mail on node WHYNOT from APON "Fred Apon" (20:25:58)

  MAIL> EXIT
```

Notes on Example 2–7:

❶ Define the logical name REVIEW, which contains commas in its string.

❷ Send a MAIL message, using REVIEW as the recipient.

❸ The sender, APON, does not receive the message. APON is the third element in a search list, and a match is found before that element is reached.

❹ Redefine the logical name REVIEW, enclosing the equivalence string in quotation marks to include the commas as part of the equivalence string.

❺ This time APON receives the message.

SYSTEM-CREATED LOGICAL NAMES

Process and Job Logical Names

Process and job logical names are available to your process. Table 2–1 describes some system-created logical names.

Table 2–1 Process and Job Logical Names Defined by the System

Logical Name	Equivalence Name
SYS$INPUT	Default input device or default file from which DCL reads input. For interactive use, SYS$INPUT is the terminal. While a command procedure is running, SYS$INPUT is the command procedure.
SYS$COMMAND	The initial file (usually your terminal) from which DCL reads input. (A file from which DCL reads input is called an input stream.) The command interpreter uses SYS$COMMAND to "remember" the original input stream.
SYS$ERROR	Default device to which the system writes messages generated by warnings and errors. For an interactive user, SYS$ERROR is the terminal.
SYS$OUTPUT	Default output device. For an interactive user, SYS$OUTPUT is the terminal. (A file to which DCL writes output is called an output stream.)
TT	Default device name for your interactive terminal.
SYS$LOGIN	Default disk and directory established at login time. Specified in the user authorization record by the system manager.
SYS$LOGIN_DEVICE	Default disk established at login time. Specified in the user authorization record by the system manager.
SYS$DISK	Default disk established at login. Changed by the SET DEFAULT command.
SYS$NET	The source process that invokes a target process in DECnet task-to-task communication. When opened by the target process, SYS$NET represents the logical link over which that process can exchange data with its partner. SYS$NET is defined only during task-to-task communication.

System Logical Names

System logical names are available to all users on the system.

Table 2–2 lists some of the system logical names commonly used by the system manager.

Table 2–2 Some of the System Logical Names Defined by the System

Logical Name	Equivalence Name
SYS$SYSDEVICE	Device on which the VMS operating system files reside
SYS$COMMON	Root directory for VMS system files shared by nodes in a cluster
SYS$SPECIFIC	Root directory for VMS system files specific to a single node in a cluster
SYS$SYSROOT	Root directory for VMS system files; points to both SYS$COMMON and SYS$SPECIFIC
SYS$MANAGER	Default device and directory for the SYSTEM account; contains some system data files such as the operator log
SYS$SYSTEM	Device and directory containing operating system programs and other system files
SYS$STARTUP	Device and directory containing command procedures that are executed when the system starts up
SYS$NODE	Network node name for the local system, if DECnet software is active on the system

Redefining System-Created Logical Names

Redefining SYS$OUTPUT

You can redefine SYS$OUTPUT to redirect output from your default device to another file.

In the following example, the display produced by SHOW DEVICES is directed to MYFILE.LIS in your default directory rather than to your terminal:

```
$ ASSIGN MYFILE.LIS  SYS$OUTPUT
$ SHOW DEVICES
$ DEASSIGN SYS$OUTPUT
```

- Remember to deassign SYS$OUTPUT, or output will continue to be written to the file you have specified.
- You can redefine SYS$OUTPUT to redirect output from an image, using the ASSIGN/USER_MODE command.
  - Once the image exits, SYS$OUTPUT resumes its default value.

```
$ ASSIGN/USER_MODE MYFILE.LIS  SYS$OUTPUT
$ SHOW DEVICES
```

DEFINING NAMES IN THE SYSTEM TABLE

Defining Logical Names Clusterwide

To define the same system logical name on every node of a cluster, you can use the SYSMAN utility.

```
$ SET DEFAULT SYS$SYSTEM
$ SET PROCESS/PRIVILEGE=SYSPRV
$ RUN SYSMAN
SYSMAN> SET ENVIRONMENT/CLUSTER
%SYSMAN-I-ENV, current command environment:
        Clusterwide on local cluster
        Username SYSTEM        will be used on nonlocal nodes

SYSMAN> DO ASSIGN /SYSTEM /TRANSLATION=CONCEALED $1$DUA9: NEWDISK
%SYSMAN-I-OUTPUT, command execution on node BARNUM
%SYSMAN-I-OUTPUT, command execution on node RNGLNG
%SYSMAN-I-OUTPUT, command execution on node LION
%SYSMAN-I-OUTPUT, command execution on node HORSE
%SYSMAN-I-OUTPUT, command execution on node BAILEY
%SYSMAN-I-OUTPUT, command execution on node BEAR
%SYSMAN-I-OUTPUT, command execution on node TIGER
SYSMAN> CTRL/Z
```

Defining Logical Names Permanently

To permanently assign a system logical name, place a DCL command to assign it in the command procedure SYS$STARTUP:SYLOGICALS.COM.

- Remember to use the appropriate attributes, such as /TRANSLATION=CONCEALED.

DURATION OF LOGICAL NAMES

A **process** or **job** logical name assignment lasts until you:

- Log out (or otherwise stop the process)
  - Job logical name assignments last until the last process in the job logs out or is stopped.
- Assign the logical name to a different string
- Remove the logical name with the DEASSIGN command

A **group** or **system** logical name assignment lasts until:

- You shut down the system
- The system fails
- You assign the logical name to a different string
- You remove the logical name with the DEASSIGN command

If you always want to assign a process or job logical name, put the assignment statement in your LOGIN.COM procedure. If you always want to assign a system logical name, put the assignment statement in SYS$STARTUP:SYLOGICALS.COM.

SUMMARY

- The system stores logical names and their equivalence strings in four default logical name tables and possibly additional user-defined tables:
  - Process
  - Job
  - Group
  - System
  - User-defined
- Two commands, SHOW LOGICAL and SHOW TRANSLATION, are available to determine the equivalence of a logical name.
- Translation attributes TERMINAL and CONCEALED are used to modify the interpretation of the equivalence of a logical name.
- The system creates some job, process, and system logical names automatically.
- Redefining some system logical names is useful for redirecting input and output.
- Process and job logical name assignments last until the user:
  - Logs out
  - Assigns the logical name to a different string
  - Removes the logical name with the DEASSIGN command
- Group and system logical names last until:
  - You shut down the system
  - The system fails
  - You assign the logical name to a different string
  - You remove the logical name with the DEASSIGN command

Queue Management

INTRODUCTION

This chapter introduces the concepts of queue management. Among the topics discussed are:

- The VMS system queue facilities
- How the VMS system handles print and batch queues
- Monitoring print and batch queues
- VAXcluster queue management

OBJECTIVES

To describe the tasks and responsibilities involved in queue management, a system and network manager should be able to:

- Describe what queue facilities are and how the VMS system uses them
- Monitor print and batch queues
- Set up and manage queues in a VAXcluster system

RESOURCES

- *VMS DCL Dictionary*
- *VMS System Manager's Manual*
- *Guide to Setting Up a VMS System*
- *Guide to Maintaining a VMS System*

TOPICS

- Queue facilities and operations
- How the VMS system handles print jobs
- How the VMS system handles batch jobs
- Batch queue operations
- Monitoring print and batch queues
- Managing batch and print operations in a VAXcluster system

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.

The Queue Manager

The queue manager process (QUEUE_MANAGER) controls print and batch queues and jobs.

- One queue manager performs these tasks for the entire cluster.
- Starting the queue manager:
  - The queue manager is automatically started when the first node of the cluster starts up (provided you have entered a START/QUEUE/MANAGER command in the past).
  - The command STOP/QUEUE/MANAGER/CLUSTER stops the queue manager.
  - The command START/QUEUE/MANAGER restarts it.
  - If the node on which the queue manager is running fails, a new queue manager automatically starts on another node in the cluster.
- The queue manager keeps track of queues and jobs in a database consisting of three files:
  - SYS$SYSTEM:QMAN$MASTER.DAT, the master file
  - SYS$SYSTEM:SYS$QUEUE_MANAGER.QMAN$QUEUES, the queue file
  - SYS$SYSTEM:SYS$QUEUE_MANAGER.QMAN$JOURNAL, the journal file

The queue manager communicates with the job controller process (JOB_CONTROL) on each system, which performs many system management and control tasks.

Types of Queues

The VMS operating system provides two general classes of queues:

- Execution
  - Accepts either batch or print jobs for processing, depending on how the queue was initialized (created)
- Generic
  - Holds jobs until they are transferred to an assigned execution queue

Queue classes are further defined by:

- The kind of job the queue accepts
- The type of device to which output is directed

Execution Queues

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

- Batch
  - Can only accept (process) batch jobs
  - Executes as a **detached** process
- Output
  - Accepts (typically) print jobs for processing by an independent process called a *symbiont*
  - Three types of output execution queues:

Printer	Directs output to line printers
Terminal	Directs output to terminal printers (printers attached to terminal lines)
Server	Processes files in the queue using a specially created symbiont

  - Symbionts for server execution queues can be customer-written or provided as part of a layered product

    Not necessarily used for print output operations
    Not covered in this course

Generic Queues

Generic queues are used to hold a job until an associated execution queue becomes available. There are two types of generic queues:

- Generic batch queue
  - Directs jobs only to batch execution queues
  - Typically used in VAXcluster systems to distribute the workload across several nodes
- Generic print queue
  - Directs jobs to any of the three types of output execution queues: printer, terminal, or server
- The list of associated execution queues is defined when the generic queue is initialized.
- When an execution queue becomes available, the job is **requeued** from the generic queue to the execution queue.

HOW THE VMS SYSTEM HANDLES PRINT JOBS

There are several ways to print on a VMS system. A printer is controlled by a print symbiont if it is associated with a queue. You can:

- Allocate a printer and send data interactively to it with the **COPY** command
- Allocate a printer and send data from a program to it with the **WRITE** command
- Use the **PRINT** command

COPY and **WRITE** have several disadvantages:

- If you have limited numbers of printers, you must wait until one can be allocated to your process.
- If you have multiple users, waiting for available printers can cause time problems.
- There is no capability to order print jobs by their importance or size; printing is on a first-allocated, first-served basis.

These problems are solved in part by print queues.

- The PRINT command places print jobs in print queues.
- The system then decides when and where to print any particular job.
  - Print queues solve waiting and scheduling problems.
  - The PRINT command causes QUEUE_MANAGER to place a job in the queue.
  - Print symbionts execute print jobs.
  - JOB_CONTROL sends jobs to print symbionts.

Example 3–1 shows the results of the SHOW SYSTEM command. Note the JOB_CONTROL and SYMBIONT_0001 processes. This node does not happen to be running the queue manager.

Example 3–1 JOB_CONTROL and Print Symbiont Processes

```
$ SHOW SYSTEM
VAX/VMS V5.5    on node BIMBAM  14-NOV-1991 17:21:45.54   Uptime  21 09:23:21
  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           CUR      4      328   0 00:00:10.49      1057    299
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
20200079 _RTA1:          HIB      6    22780   0 00:20:36.65     10459   4096
```

Notes on Example 3–1:

❶ The JOB_CONTROL process

❷ A print symbiont process

Print Job Scheduling

After jobs are placed in the queue using the PRINT command:

QUEUE_MANAGER process schedules the jobs using:

- Priority
  - — Job with highest queue priority executed first
  - — Priority of jobs in queues limited by two system parameters:

    SYSGEN

    DEFQUEPRI — The default queue priority assigned to all print jobs (100)

    MAXQUEPRI — The maximum queue priority any user can assign to a job (range is 0-255)
  - — Job size

    Smaller jobs executed before larger jobs (within the same priority group)
  - — Submission time

    Jobs executed in order of submission if they are same size and have same priority

Scheduling can be changed to first-come-first-served with the qualifier **/SCHEDULE=NOSIZE** on the INITIALIZE/QUEUE or SET QUEUE command.

Example 3–2 illustrates how print jobs are scheduled by the QUEUE_MANAGER process.

Example 3–2 Scheduling Print Jobs

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

  Entry  Jobname         Username     Blocks  Status
  -----  -------         --------     ------  ------
❶   228  ACTION          JONES        6       Printing
    Submitted 13-DEC-1991 12:02 /FORM=DEFAULT /PRIO=100
      _DRA1:[JONES]ACTION.COM;1 /COPIES=2

    231  NOTES           JONES        12      Pending
    Submitted 13-DEC-1991 12:15 /FORM=DEFAULT /PRIO=120
      _DRA1:[JONES]NOTES.TXT;1                    ❷
                                           ❸
    230  MEMO            JONES        1       Pending
    Submitted 13-DEC-1991 12:08 /FORM=DEFAULT /PRIO=100
      _DRA1:[JONES]MEMO.MEM;1

❹   229  MATH            JONES        6       Pending
    Submitted 13-DEC-1991 12:04 /FORM=DEFAULT /PRIO=100
      _DRA1:[JONES]MATH.LIS;1
```

Notes on Example 3–2:

❶ Job 228 is currently executing. The QUEUE_MANAGER process examines the parameters of the pending jobs to determine which job to 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 jobs**
- **MAXQUEPRI — the maximum queue priority any user can assign to a job (range is 0-255).**

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 QUEUE_MANAGER process examines the parameters of Job 229 and the new job to determine which job to print first.

Creating Print Queues

Establishing a print execution queue requires OPER privilege. To establish a print execution queue:

1. Set physical attributes of device
   - **SET PRINTER**
   - **SET TERMINAL** attributes
2. Spool each printing device
   - **SET DEVICE/SPOOLED**
3. Initialize and start an execution queue for each printing device
   - /NOENABLE_GENERIC to prevent system from automatically moving jobs
4. Initialize and start a generic print queue
   - Normally **SYS$PRINT**
5. Initialize and start a generic terminal queue
   - Use any name, TERMPRINT, for example:

```
$ INITIALIZE/QUEUE/TERMINAL/GENERIC=(TTA1,TTC7) TERMPRINT
```

Queues can be created either:

- Interactively

  or

- In a command procedure

You can create and start queues using one or more commands as shown in Table 3–1.

Table 3–1 Initializing and Starting Queues

Command	Comments

`$ 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 execute under the new parameters.

`$ START/QUEUE [/qualifiers] queue-name` `$ START/QUEUE 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 procedure SYSTARTUP_V5.COM. If the queue is already running, this command has no effect.

Table 3–2 lists commands that create and use print execution queues.

Table 3–2 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`	Lists the devices and selects one.

Set the device attributes	`$ SET PRINTER -`
	`_$ /UPPER LPA0`
		`$ SET TERMINAL -`
		`_$ /PERMANENT -`
		`_$ /NOTYPE_AHEAD -`
		`_$ /SPEED=2400 -`
		`_$ /NOBROADCAST -`
		`_$ TTA3`
			Sets the attributes of the 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 -`
	`_$ LINE_PRINTER`
		`$ INITIALIZE/QUEUE -`
		`_$ /TERMINAL -`
		`_$ /START/ON=TTA3 -`
		`_$ LINE_PRINTER`
			Assigns the queue a different name than its device name if desired by using the **/ON** qualifier. (By default, the name of the queue matches the name of the printer.)

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 is similar, but the SYS$PRINT queue is defined to print on TTA3.

Creating Generic Print Queues

Establish generic print queues when you have more than one printer set up in the same fashion, and want to share the processing among the printers.

- The system does not move jobs from generic queues to execution queues initialized with **/NOENABLE_GENERIC** qualifier.
- Execution queues are given the **/ENABLE_GENERIC** attribute by default.

Table 3–3 shows the steps used in creating and using generic print queues.

Table 3–3 Creating and Using Generic Print Queues

Operation	Command	Comment

Creates an execution queue for a printer	`$ SET PRINTER/UPPER LPA0`
	`$ SET DEVICE/SPOOLED LPA0`
	`$ INITIALIZE/QUEUE/START -`
	`_$ LPA0`
		For example, the printer device LPA0

Creates an execution queue for another printer	`$ SET PRINTER/UPPER LPB0`
	`$ SET DEVICE/SPOOLED LPB0`
	`$ INITIALIZE/QUEUE/START -`
	`_$ LPB0`
		For example, the printer device LPB0

Creates a generic print queue	`$ INITIALIZE/QUEUE/START -` `_$ /GENERIC SYS$PRINT`	This queue receives default print jobs and dispense them to any execution print queues that do not use the /NOENABLE_ GENERIC qualifier.

Creates a generic print queue with specific execution queues	`$ INITIALIZE/QUEUE/START -`
	`_$ /GENERIC=(LPA0,LPB0) -`
	`_$ SYS$PRINT`
		This queue receives default print jobs and dispense them to LPA0 or LPB0.

Uses 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.

Example 3–3 shows queue status display of current, pending, and holding jobs.

Example 3–3 Queue Status Display of Current, Pending, and Holding Jobs

```
$ SHOW QUEUE/DEVICE/GENERIC
Generic queue AFTER5, assigned to LPC0

  Entry  Jobname          Username     Blocks  Status
  -----  -------          --------     ------  ------
    497  LATER            JONES             1  Pending

Printer queue FORM3, stopped

  Entry  Jobname          Username     Blocks  Status
  -----  -------          --------     ------  ------
    389  MATH             JONES             5  Pending
    320  TEST             JONES             7  Pending

Printer queue LPA0

  Entry  Jobname          Username     Blocks  Status
  -----  -------          --------     ------  ------
    492  MEMO             JONES             1  Printing

Printer queue LPB0

  Entry  Jobname          Username     Blocks  Status
  -----  -------          --------     ------  ------
    493  ACTION           JONES             1  Printing
    496  TABLES           JONES            21  Pending

Printer queue LPC0

  Entry  Jobname          Username     Blocks  Status
  -----  -------          --------     ------  ------
    494  FORTEST          JONES             1  Printing

Printer queue OVERNIGHT, stopped

  Entry  Jobname          Username     Blocks  Status
  -----  -------          --------     ------  ------
    419  LONG             JONES           400  Pending
    411  BIGJOB           JONES           478  Pending

Generic printer queue SYS$PRINT

  Entry  Jobname          Username     Blocks  Status
  -----  -------          --------     ------  ------
    495  MEMO             JONES             1  Pending

Terminal printer queue TERM, on TTA3:

  Entry  Jobname          Username     Blocks  Status
  -----  -------          --------     ------  ------
    491  PROG             JONES            10  Printing
    498  NOTES            JONES             9  Pending
$
```

Automatic Queue Creation

To invoke automatic queue creation at boot time:

- Include queue commands in the startup command procedure to start queues at system startup.
- It is better to include a line in the startup command procedure to invoke a separate command procedure to start up queues.

System shutdown procedure stops all queues (SYS$SYSTEM:SHUTDOWN.COM).

Example 3–4 illustrates startup commands in SYSTARTUP_V5.COM.

Example 3–4 Startup Commands in SYSTARTUP_V5.COM

```
$ SET NOON
$ !
$ ! 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  LPB0
$ !
$ 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
$
```

Monitoring Print Queues and Jobs

You can monitor the status of print queues. The amount of information displayed depends on your privileges and queue ownership rights.

Monitoring Print Queues

- Use the **SHOW QUEUE** command to monitor an entire queue:

  ```
  $ SHOW QUEUE [/qualifiers] [queue-name]
  $ SHOW QUEUE/SUMMARY/DEVICE=(PRINTER,TERMINAL)
  ```

- Default action is to display status of all queues and all jobs owned by you.
- Queues are displayed in alphabetical order.
- Qualifiers provide selection of the type and amount of queue information to be displayed.
- Queue status codes indicate current state of the queue (see Table 3–6).

The SHOW QUEUE qualifiers allow you to select the type of queue information you want to display (see Table 3–4), or the amount of information you want to display (see Table 3–5).

Table 3–4 SHOW QUEUE Qualifiers for Displaying Types of Queues

Qualifier	Description
`/BY_JOB_STATUS=status-type`	Displays queues that contain jobs of a specified type of status. If no keyword is specified, the jobs of all status-types are displayed. The types are EXECUTING, HOLDING, PENDING, RETAINED, and TIMED_RELEASE.
`/BATCH`	Displays the status of batch execution queues.
`/DEVICE=keyword-list`	Displays particular type of queue: PRINTER, SERVER, or TERMINAL. If no keywords are specified, all types of output queues are displayed.
`/GENERIC`	Displays the status of generic queues.

Table 3–5 SHOW QUEUE Qualifiers for Displaying the Amount of Queue Information

Qualifier	Description
`/ALL_JOBS` or `/ALL_ENTRIES`	Displays information about all jobs or entries 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.
`/FILES`	Adds the list of files associated with each job to the display.
`/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.

Table 3–6 shows queue status codes.

Table 3–6 Queue Status Codes

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.
`Operator service`	A **PRINT/OPERATOR** command has been executed.
`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.
`Stalled`	Print symbiont processing is temporarily halted due to a device-related problem.
`Starting`	Queue has been started, but the print symbiont process is not yet active.
`Stop pending`	Queue will be **stopped** when current jobs have finished executing.
`Stopped`	A **STOP/QUEUE** command specified with either a /**NEXT**, **REQUEUE**, or **RESET** qualifier has been executed.
`Stopping`	The queue manager is processing a **STOP/QUEUE** command specified with either a /**NEXT**, **REQUEUE**, or **RESET** qualifier.

Monitoring Print Jobs

Use the **SHOW ENTRY** command to monitor individual jobs:

```
$ SHOW ENTRY [/qualifiers] [entry-number or job-name]
$ SHOW ENTRY 228
$ SHOW ENTRY ACTION
$ SHOW ENTRY/USER=JONES
```

Most **SHOW QUEUE** qualifiers can be used to select the type and amount of queue information to be monitored.

Queue status codes indicate current state of the job.

Table 3–7 lists common job status codes.

Table 3–7 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.
`Stalled`	Job is executing on a print queue that is stalled.
`Waiting`	Symbiont refuses the job.

Setting Print Queue Attributes

The VMS system assigns certain default attributes to each queue when you create it, such as:

- 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

See Table 3–8 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 3–8 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**. Not all parameters can be changed while the queue is running. See the output from HELP SET QUEUE for a list of parameters that can be changed.

Example 3–5 shows how to modify a running queue.

Example 3–5 Modifying a Running Queue

```
$ 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
$
```

Specifying Separation Pages

Separation pages are used to delineate between individual jobs and files within jobs.

- Job separation pages
- File separation pages

Defaults can be set for separation pages on a queue (system default is **no separation pages**).

Separation page attributes can be viewed with **SHOW QUEUE/FULL** (See Example 3–6).

Example 3–6 SHOW QUEUE — Job and File Separation Page Defaults

```
$ SHOW QUEUE/FULL LPC0
Printer queue LPC0
    /BASE_PRIORITY=4 /DEFAULT=(FLAG)
    /FORM=DEFAULT Lowercase /OWNER=[SYSTEM] /PROTECTION=(S:E,O:D,G:R,W:W)
    /SEPARATE=(BURST,FLAG,TRAILER)
$
```

- Use **/SEPARATE=option** for job separation pages (See Table 3–9).
- Use **/DEFAULT=option** for file separation pages
- Users **can** override defaults set for file separation pages
- Users **cannot** override defaults set for job separation pages

Print Order of Pages

The order of printed pages when all file and job separation page defaults are set is:

1. File burst page (/DEFAULT=BURST)
2. File flag page (/DEFAULT=FLAG) (See Figure 3–1)
3. File contents are printed
4. File trailer page (/DEFAULT=TRAILER) (See Figure 3–2)
5. Job burst page (/SEPARATE=BURST)
6. Job flag page (/SEPARATE=FLAG) (See Figure 3–3)
7. System repeats previous four steps until all files in job are printed
8. Job trailer page (/SEPARATE=TRAILER) (See Figure 3–4)

Table 3–9 lists separation page options for the /SEPARATE qualifier.

Table 3–9 Job Separation Page Options for the /SEPARATE Qualifier

Option	Description
`[NO]BURST`	Specifies a copy of the flag page 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 printed automatically with the burst page.
`[NO]FLAG`	Specifies that a page is printed preceding the job with the name of the user printed in large letters.
`[NO]TRAILER`	Specifies that a single summary sheet is printed following a job, with the name of the user printed in large letters.

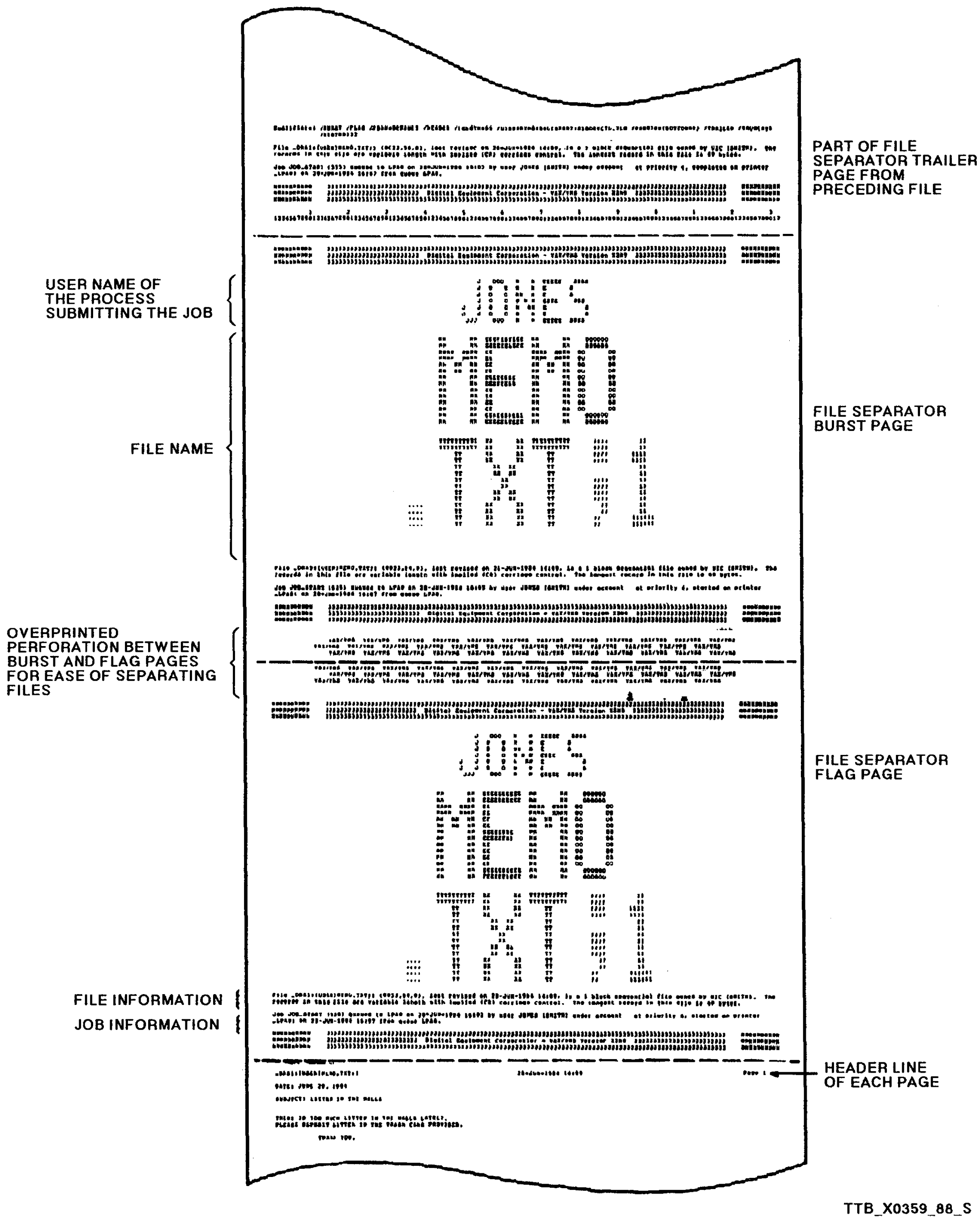
Table 3–10 lists separation page options for the /DEFAULT qualifier.

Table 3–10 File Separation Page Options for the /DEFAULT Qualifier

Option	Description
`[NO]BURST[=keyword]`	Specifies whether file burst pages are 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 printed automatically with the burst page.
`[NO]FLAG[=keyword]`	Specifies whether file flag pages are 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.
`[NO]TRAILER[=keyword]`	Specifies whether file trailer pages are 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.

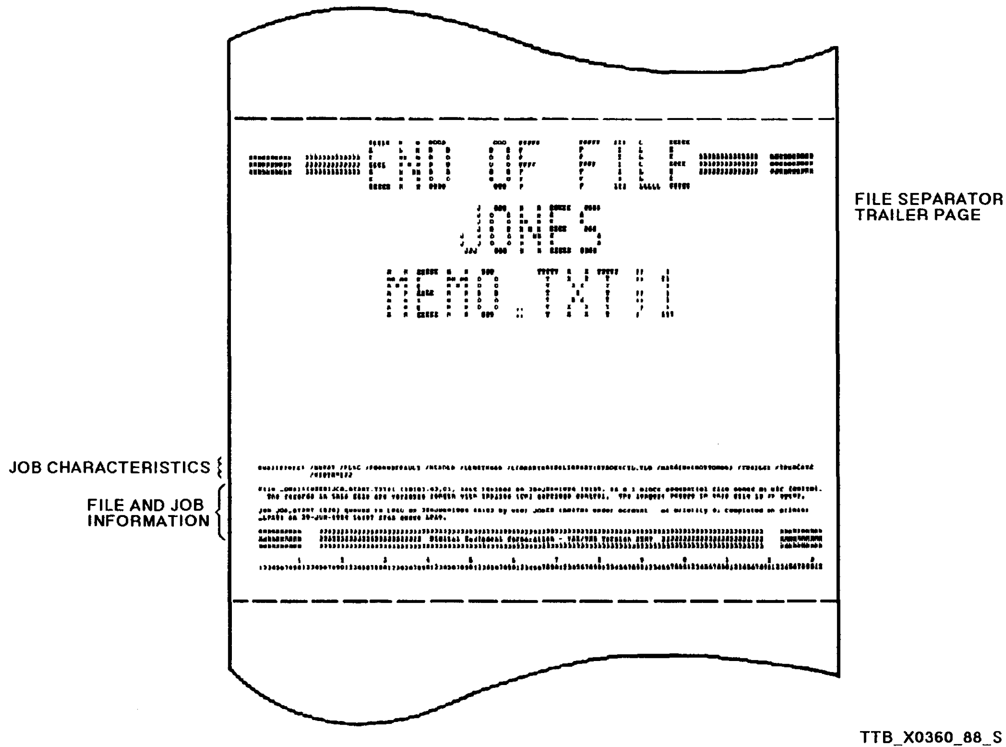
File separation burst and flag pages, specified with the /DEFAULT=BURST qualifier, are shown in Figure 3–1.

Figure 3–1 File Separation Burst and Flag Pages



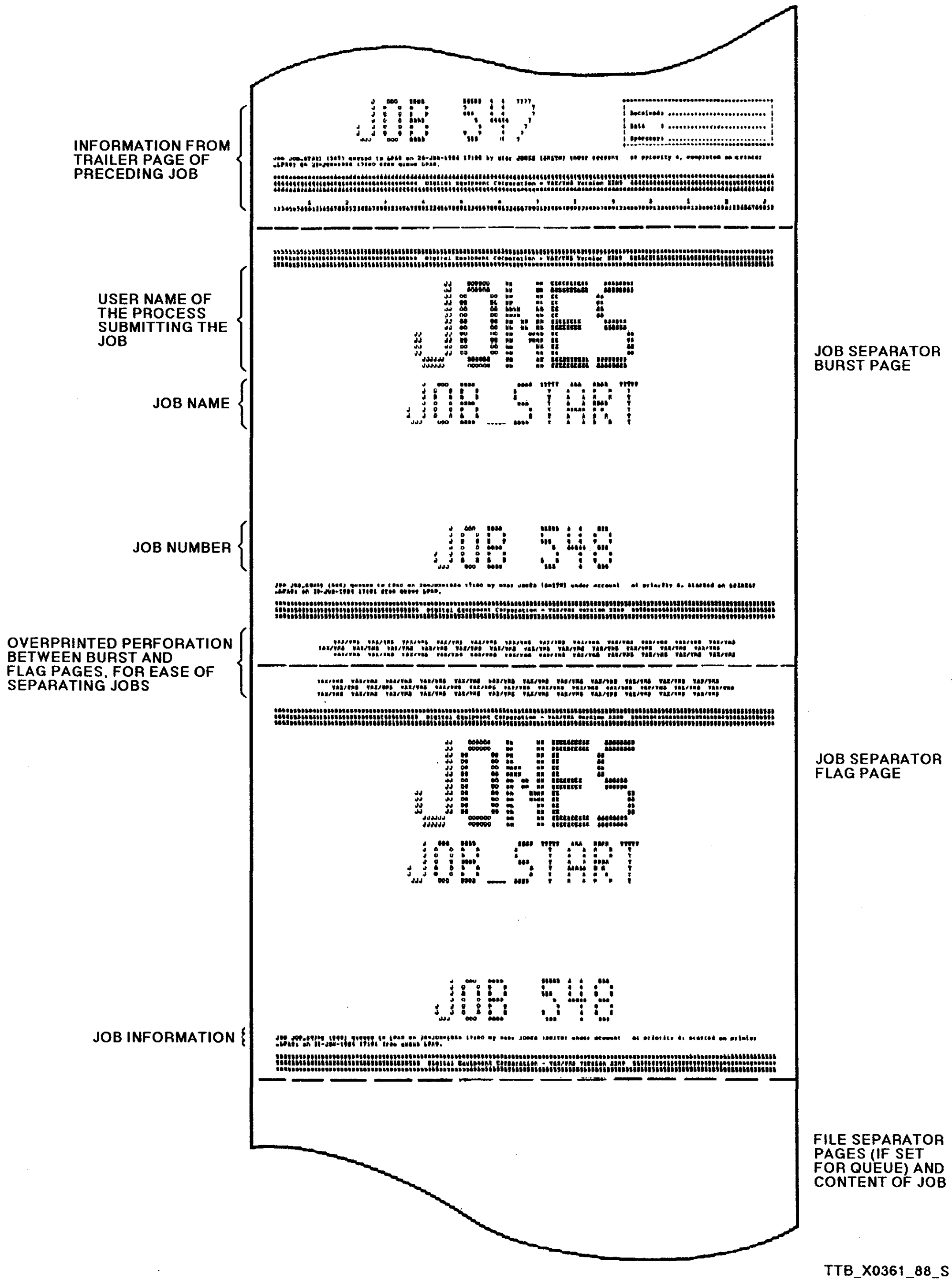
A file separation trailer page, specified with the /DEFAULT=TRAILER qualifier, is shown in Figure 3–2.

Figure 3–2 File Separation Trailer Page



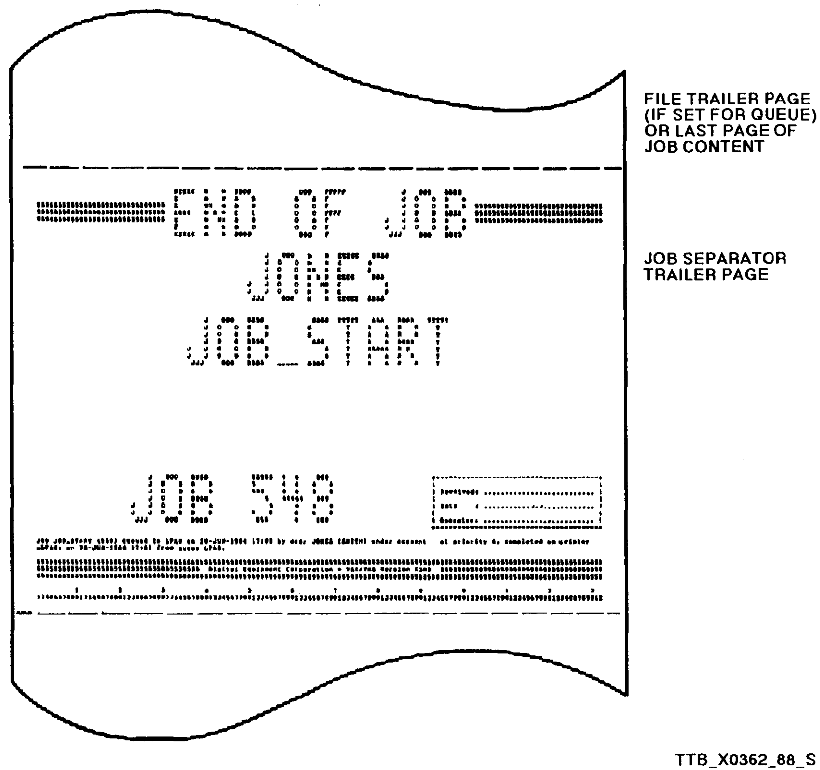
Job separation burst and flag pages, specified with the /SEPARATE=BURST qualifier, are shown in Figure 3–3.

Figure 3–3 Job Separation Burst and Flag Pages



A file separation trailer page, specified with the /SEPARATE=TRAILER qualifier, is shown in Figure 3–4.

Figure 3–4 Job Separation Trailer Page



Preventing Jobs from Being Entered in a Queue

Use **SET QUEUE/CLOSE** to prevent jobs from being entered in a queue.

Use **SET QUEUE/OPEN** to allow jobs to be entered in a queue.

Moving Jobs from One Queue to Another

- Stop a queue with **STOP/QUEUE/NEXT** before moving jobs so that no more jobs will be executed.
- Requeue the current job if the queue is an execution queue.
- Use **SET QUEUE/CLOSE** to keep additional jobs from being entered in the queue, if desired.

Example 3–7 shows commands used to move jobs from one queue to another.

Example 3–7 Moving Jobs from One Queue to Another

```
$ STOP/QUEUE LPA0
$ STOP/QUEUE/REQUEUE=LPB0 LPA0
$ ASSIGN/MERGE LPB0 LPA0
```

Deleting a Queue

You must use **STOP/QUEUE/NEXT** before deleting a queue with **DELETE/QUEUE**.

Example 3–8 shows how to delete a queue.

Example 3–8 Deleting a Queue

```
$ STOP/QUEUE/NEXT LPA0
$ DELETE/QUEUE LPA0
```

Deleting Jobs in Queues

You can delete jobs using **DELETE/ENTRY** when:

- A queue is running or when a queue is stopped
- Restarting a queue

Example 3–9 shows how to delete jobs in queues.

Example 3–9 Deleting Jobs in Queues

```
$ DELETE/ENTRY=715
$ DELETE/ENTRY=(821,823,824)
```

Handling Print Queue Problems

Problems can arise with printers:

- Paper jams
- Paper runs out
- Ribbon tears or poor print density

You might have to reprint part or all of any job that was printing when such a problem occurs. The commands needed to reprint problem jobs can also be used to set up a printer for repetitive or sequential print situations.

Use positioning and alignment qualifiers to handle these situations as illustrated in Table 3–11 and Table 3–12.

Table 3–11 Positioning a Print Job

Qualifier to **START/QUEUE**	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 3–12 Aligning Printer Paper

Command Format/Examples	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.

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.

Batch queues make waiting unnecessary and allow best use of terminals and other resources.

The **SUBMIT** command places a batch job in a batch queue.

- Batch queues must exist for the VMS system to execute batch jobs.

The QUEUE_MANAGER process schedules batch jobs to execute, and the JOB_CONTROL process creates a batch process in which to execute each job.

- Name of process composed of word BATCH and job's queue entry number
- Jobs listed in queue as currently executing
- Batch process appears in **SHOW SYSTEM** display with B in the rightmost column

Example 3–10 JOB_CONTROL and Batch Job Processes

```
    $ SHOW SYSTEM
    VAX/VMS V5.5   on node BIMBAM  14-NOV-1991 17:21:45.54   Uptime  21 09:23:21
      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
    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
(1) 202001B8 BATCH_103       CUR      3       80   0 00:00:02.93        588    259  B (2)
    2020002B AUDIT_SERVER    HIB     10       54   0 00:00:54.84       1300    223
(3) 2020002C 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
    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
```

Notes on Example 3–10:

(1) Batch process

(2) The letter B indicates that this process is executing a batch job

(3) The JOB_CONTROL process

Example 3–11 lists current and pending jobs on a batch queue.

Example 3–11 Current and Pending 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)

  Entry  Jobname         Username             Status
  -----  -------         --------             ------
    223  ACTION          JONES                Executing
❶   Submitted 13-DEC-1991 12:44 /PRIORITY=100
      _DJA0:[JONES]ACTION.COM;2 (executing)

    230  MATH            JONES                Executing
    Submitted 13-DEC-1991 12:57 /PRIORITY=100
      _DJA0:[JONES]MATH.COM;1 (executing)

❷   237  COMPUTE         JONES                Pending
    Submitted 13-DEC-1991 13:41 /PRIORITY=120
      _DJA0:[JONES]COMPUTE.COM;7 (pending)

❸   236  ACTION          JONES                Pending
    Submitted 13-DEC-1991 13:10 /PRIORITY=100
      _DJA0:[JONES]ACTION.COM;3 (pending)
$
```

Notes on Example 3–11:

❶ Jobs 223 and 230 are currently executing. The JOB_LIMIT attribute of the queue has a value of 2, limiting the number of concurrent batch processes from this queue to 2. 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 QUEUE_MANAGER process examines the parameters of the pending jobs to determine which job to execute next.

❷ Since Job 237 has a higher queue priority than Job 236, (see note 2 of Example 3–2) it will execute next.

NOTE

Although the queue priority of a batch job helps to determine when it is scheduled, the queue priority does not affect the base priority of the batch process. The base priority of batch processes is defined by:

- **The value of the BASE_PRIORITY queue attribute; in this case, 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 will execute. However, if a user submits another job before Job 236 begins executing, the QUEUE_MANAGER process compares the parameters of the new job with those of Job 236 to determine which job to execute next.

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:

- Creating batch queues
- Stopping batch queues

Creating Batch Queues

To create the default batch queue **SYS$BATCH**:

```
$ INITIALIZE/QUEUE/BATCH/START SYS$BATCH
```

- Default queue for **SUBMIT** command
- Default queue for spooled jobs from card readers

You must create all batch queues (none are automatically created).

Table 3–13 lists sample batch queues and parameter values. Table 3–14 shows initialization of batch queues.

Table 3–13 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.	Sets the owner UIC of the queue to [360,000]. Gives GROUP users READ and WRITE access only. Gives WORLD users no access. Sets larger working set extent, CPU limits, and priority. Possibly increases the job limit.
FASTQUE Used by all who need a job done quickly.	Sets the base priority at 5. Uses default queue protection or possibly restricts use to a certain group, UIC, or ACL. Sets job limit at 2, but limits the maximum CPU to a low value to keep the queue from taking over the system. Optionally uses higher working set limits.
SLOWQUE Used by all who want to run a batch job that affects system performance as little as possible.	Sets the priority at 3. DO NOT set to 0 or 1, as jobs may get very little CPU time and finish too slowly. Uses default queue protection. Sets job limit to 1. Uses default working set limits.
ZOOMQUE Very fast queue that you only start after hours or during lunch.	Sets the base priority at 6 or higher, but definitely sets a low CPU limit. Uses default queue protection. Sets job limit to 2. Sets high working set limits. You should note, however, that setting a batch queue's default BASE_PRIORITY to a value higher than the normal interactive value is generally considered somewhat dangerous to system response time. You should carefully monitor any batch queue running in this fashion to avoid system degradation.
CADCAM Used by large, compute-intensive applications, such as engineering, manufacturing, or modeling programs.	Sets up as shown with SLOWQUE, but increases working set limits to large values. Sets CPU limit to very large value, or infinite. Optionally protects the queue for access by only specific user groups.

Table 3–14 Qualifiers to INITIALIZE/QUEUE for Batch Queues

Qualifiers	Examples	Comments
/JOB_LIMIT	`$ INITIALIZE/QUEUE/BATCH -` `_$ /JOB_LIMIT=2 FASTBAT`	Sets a limit on the number of batch processes that can run concurrently from one batch queue. The default is 1.
/BASE_PRIORITY	`$ INITIALIZE/QUEUE/BATCH -` `_$ /BASE_PRIO=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 -`
	`_$ /OWNER=[ENG,PROJ5] -`
	`_$ /PROT=(S:E,O:D,G:R,W:W)`
		Limits access to a queue.

/WSDEFAULT
/WSQUOTA
/WSEXTENT
/DISABLE_
SWAPPING
	`$ INITIALIZE/QUEUE/BATCH -`
	`_$ /WSDEFAULT=500 -`
	`_$ /WSQUOTA=800 -`
	`_$ /WSEXTENT=2000 -`
	`_$ /DISABLE_SWAPPING -`
	`_$ 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).

/CPUMAXIMUM	`$ INITIALIZE/QUEUE/BATCH -`
	`_$ /CPUMAXIMUM=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.

/CPUDEFAULT	`$ INITIALIZE/QUEUE/BATCH -`
	`_$ /CPUDEFAULT=00:03:00 -`
	`_$ 3MINBAT`
		Sets default CPU limit assigned to batch processes from this queue. Otherwise, the default is the CPU limit in the user's UAF record or the value of **/CPUMAXIMUM** for the queue.

Stopping Batch Queues

To stop execution of the current job:

```
$ STOP/QUEUE
```

To complete current job before stopping queue:

```
$ STOP/QUEUE/NEXT
```

To stop a job executing in a queue:

```
$ STOP/ENTRY=job_number
$ STOP/ENTRY=205
```

Example 3–12 shows how to stop batch queues.

Example 3–12 Stopping Batch Queues

```
❶ $ SUBMIT ACTION.COM
  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

    Entry  Jobname          Username              Status
    -----  -------          --------              ------
      911  ACTION           JONES                 Executing
  $
  $ SHOW SYSTEM/BATCH
  VAX/VMS V5.5  on node BIMBAM 13-DEC-1991 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

    Entry  Jobname          Username              Status
    -----  -------          --------              ------
      911  ACTION           JONES                 Executing
  $
  $ SHOW SYSTEM/BATCH
  VAX/VMS V5.5  on node BIMBAM 13-DEC-1991 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
  $
```

Notes on Example 3–12:

❶ The command file 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 the system indicates the executing batch job as being in a **suspended** (SUSP) state.

❹ Starts the SYS$BATCH batch queue running again.

❺ Informs the queue manager to stop the SYS$BATCH queue after the current batch job is 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 remain in a **pending** state until the queue is restarted.

❼ Note 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.

MANAGING BATCH AND PRINT OPERATIONS IN A VAXcluster SYSTEM

Distributed Queuing

The job controller and queue manager distribute the batch and print processing workload over cluster members:

- Permit users to submit batch and print jobs to queues that execute on any node in the cluster
- Use a common queue database to maintain the current state of all queues on all systems on the cluster
- Allow generic queues to be created that feed execution queues on any systems in the cluster
- Direct batch and print jobs to the execution queue with the lowest ratio of jobs to queue limit (or to the next available queue)
- Use the distributed lock manager to signal other members to examine the batch and print queues for jobs to be processed

Setting Up Cluster-Wide Queues

- Queue database stores cluster-wide queue information on a cluster-available disk.
- You must make queue names unique to avoid conflicts in the cluster-wide queue database.

Setting Up Cluster-Wide Batch and Print Queues

- Initialize queues from each node where they will be used
- Start each queue on the node where its print or CPU resource is located
- Include the /ON=resource_name qualifier to specify the exact name of the printer or CPU as needed
- Resource_name is of the form:
  - **node::** for a batch queue
  - **node::device:** for a print queue

Examples of Creating Cluster-Wide Generic Print and Batch Queues

Example 3–13 Queue Creation Commands on BARNUM

```
$ INITIALIZE/QUEUE/ON=BARNUM::LPA0:/START   BARNUM_PRINT
$ INITIALIZE/QUEUE/ON=BAILEY::LPA0:         BAILEY_PRINT
$ INITIALIZE/QUEUE/GENERIC=(BARNUM_PRINT,BAILEY_PRINT) -
  /START CLUSTER_PRINT
$ DEFINE/SYSTEM SYS$PRINT CLUSTER_PRINT
```

Example 3–14 Queue Creation Commands on BAILEY

```
$ INITIALIZE/QUEUE/ON=BARNUM::LPA0:        BARNUM_PRINT
$ INITIALIZE/QUEUE/ON=BAILEY::LPA0:/START  BAILEY_PRINT
$ INITIALIZE/QUEUE/GENERIC=(BARNUM_PRINT,BAILEY_PRINT) -
  /START CLUSTER_PRINT
$ DEFINE/SYSTEM SYS$PRINT CLUSTER_PRINT
```

Example 3–15 Queue Creation Commands on All Other Cluster Members

```
$ INITIALIZE/QUEUE/ON=BARNUM::LPA0:        BARNUM_PRINT
$ INITIALIZE/QUEUE/ON=BAILEY::LPA0:        BAILEY_PRINT
$ INITIALIZE/QUEUE/GENERIC=(BARNUM_PRINT,BAILEY_PRINT) -
  /START CLUSTER_PRINT
$ DEFINE/SYSTEM SYS$PRINT CLUSTER_PRINT
```

Example 3–16 SHOW QUEUE Output on Any Cluster Member

```
$ SHOW QUEUE/DEVICE/FULL
Printer queue BARNUM_PRINT, on BARNUM::LPA0:
/BASE_PRIORITY=4 /DEFAULT=(FEED) /FORM=DEFAULT
/OWNER=[SYSTEM] /PROTECTION=(S:E,O:D,G:R,W:W)

Printer queue BAILEY_PRINT, on BAILEY::LPA0:
/BASE_PRIORITY=4 /DEFAULT=(FEED) /FORM=DEFAULT
/OWNER=[SYSTEM] /PROTECTION=(S:E,O:D,G:R,W:W)

Generic printer queue CLUSTER_PRINT
/GENERIC=(BARNUM_PRINT,BAILEY_PRINT) /OWNER=[SYSTEM]
/PROTECTION=(S:E,O:D,G:R,W:W)
```

Creating a Cluster-Wide Generic Batch Queue

Example 3–17 Creating and Displaying Cluster-Wide Batch Queues

On LION:

```
$ INITIALIZE/QUEUE/BATCH/ON=LION::/START   LION_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=TIGER::        TIGER_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=BEAR::         BEAR_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=HORSE::        HORSE_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=BARNUM::       BARNUM_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=RNGLNG::       RNGLNG_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=BAILEY::       BAILEY_BATCH
$ INITIALIZE/QUEUE/BATCH/GENERIC=(LION_BATCH,TIGER_BATCH,BEAR_BATCH,-
  HORSE_BATCH,BARNUM_BATCH,RNGLNG_BATCH,BAILEY_BATCH)/START  SYS$BATCH
```

On BAILEY:

```
$ INITIALIZE/QUEUE/BATCH/ON=LION::         LION_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=TIGER::        TIGER_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=BEAR::         BEAR_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=HORSE::        HORSE_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=BARNUM::       BARNUM_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=RNGLNG::       RNGLNG_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=BAILEY::/START BAILEY_BATCH
$ INITIALIZE/QUEUE/BATCH/GENERIC=(LION_BATCH,TIGER_BATCH,BEAR_BATCH,-
  HORSE_BATCH,BARNUM_BATCH,RNGLNG_BATCH,BAILEY_BATCH)/START  SYS$BATCH
```

On any cluster member:

```
$ SHOW QUEUE/BATCH/FULL
Batch queue HORSE_BATCH, on HORSE::
/BASE_PRIORITY=4 /JOB_LIMIT=1 /OWNER=[SYSTEM]
/PROTECTION=(S:E,O:D,G:R,W:W)
    .
    .
    .
Batch queue BAILEY_BATCH, on BAILEY::
/BASE_PRIORITY=4 /JOB_LIMIT=1 /OWNER=[SYSTEM]
/PROTECTION=(S:E,O:D,G:R,W:W)

Generic batch queue SYS$BATCH
/GENERIC=(LION_BATCH,TIGER_BATCH,BEAR_BATCH,
HORSE_BATCH,BARNUM_BATCH,RNGLNG_BATCH,BAILEY_BATCH)
/OWNER=[SYSTEM] /PROTECTION=(S:E,O:D,G:R,W:W)
```

Common Queue Startup

Example 3–18 Common Command Procedure for Queue Startup

```
$ SET NOON
!
! STARTQUE.COM for all nodes
!
! Initialize Symbols
$ LION_START="/NOSTART"
$ TIGER_START="/NOSTART"
$ BEAR_START="/NOSTART"
$ HORSE_START="/NOSTART"
$ BARNUM_START="/NOSTART"
$ RNGLNG_START="/NOSTART"
$ BAILEY_START="/NOSTART"
!
! Set symbol for this member
!
$ NODE = F$GETSYI("NODENAME")
$ 'NODE'_START = "/START"
!
!
! Initialize and start cluster print queues. These are the only two members
! that have local printers.
!
$ INIT/QUE/ON=BARNUM::LPA0:'BARNUM_START BARNUM_PRINT
$ INIT/QUE/ON=BAILEY::LPA0:'BAILEY_START BAILEY_PRINT
!
! Initialize and start the cluster-wide generic print queue. All members in
! the cluster need to start this generic queue.
!
$ INIT/QUE/GENERIC=(BARNUM_PRINT,BAILEY_PRINT)/START SYS$PRINT
!
! Initialize and start batch queues
$ INITIALIZE/QUEUE/BATCH/ON=BARNUM::'BARNUM_START  BARNUM_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=LION::'LION_START      LION_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=TIGER::'TIGER_START    TIGER_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=BEAR::'BEAR_START      BEAR_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=HORSE::'HORSE_START    HORSE_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=RNGLNG::'RNGLNG_START  RNGLNG_BATCH
$ INITIALIZE/QUEUE/BATCH/ON=BAILEY::'BAILEY_START  BAILEY_BATCH
!
! Initialize and start the cluster-wide generic batch queue.
!
$ INITIALIZE/QUEUE/BATCH/GENERIC=(LION_BATCH,TIGER_BATCH,BEAR_BATCH,-
  HORSE_BATCH,BARNUM_BATCH,RNGLNG_BATCH,BAILEY_BATCH)/START  SYS$BATCH
```

SUMMARY

There are two types of queues:

- Execution
- Generic

Overview of Queue Commands

Most queue commands require either OPER privilege or Execute (E) access to the target queue.

Table 3–15 summarizes the various classes of DCL queue commands and their usage.

Table 3–15 Summary of 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
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

Performing Backups and Restores

INTRODUCTION

This chapter presents the basic concepts of performing backups and restores. Among the topics covered are:

- The method used for making backup copies of system files
- How to use command procedures for backups
- How to restore files from backup copies
- How to list the contents of a save set

OBJECTIVES

To perform backups and restores, a system and network manager should be able to:

- Make image and incremental backup copies of files
- Use command procedures for backups
- Restore files from Image and Incremental backup copies
- List the contents of a save set

RESOURCES

- *VMS DCL Dictionary*
- *VMS System Manager's Manual*
- *Guide to Setting Up a VMS System*
- *Guide to Maintaining a VMS System*

TOPICS

- Making backup copies of files
- Using command procedures for backups
- Restoring files from backup copies
- Listing the contents of a save set

MAKING BACKUP COPIES OF FILES

You should make backup copies of the files on your system on a regular and scheduled basis.

There are two general types of backups that save copies of all of the files on your system:

- Image (full) backups
- Incremental backups

Image and Incremental Backups

You can make either image backups (also called full backups) or incremental backups.

- Image Backups—save a copy of all files
  - Easier and faster to restore than an incremental backup
- Incremental Backups—save a copy of those files that have been created or modified since the most recent backup
  - Useful only when an image backup using the /RECORD qualifier has previously been taken
  - Performed more quickly than image backups
  - Require less storage space
- Files that have been backed up with a combination of image and incremental backups are more complicated to restore than files that have been backed up in an image backup.
- If you choose to use incremental backup, remember that you must also make periodic image backups of your files.
- It might be reasonable to make an image backup weekly and to make daily incremental backups for disks where files are updated frequently.
- For small systems, it can be useful to use a batch job that backs up the entire system daily, resubmitting itself automatically.
- When you perform any type of backup, it is best to notify users using the REPLY/ALL command. An example of a backup notification using REPLY/ALL follows:

  ```
  $ REPLY/ALL "System Backup about to begin"
  ```

Save Sets

When you back up files in a save operation, the files are stored in units called **save sets**.

A save set is a file created and used by Backup when you use the BACKUP command to save files.

The save set includes the files that you save with Backup and other information that is used by Backup.

A save set can be:

- A file on tape
- A disk file on the local system
- A disk file on another system in the network

Using the BACKUP Command to Save Files

The BACKUP command line has three key parts:

```
$ BACKUP/qualifiers  file_specification/qualifiers  saveset_specification/qualifiers
```

- BACKUP/qualifiers—The BACKUP command and its qualifiers
- file_specification/qualifiers—gives information about the disk and files that currently exist and are to be backed up
- saveset_specification/qualifiers—this part of the command line identifies the tape drive (or disk or diskette), optionally the node, and the save-set name to which files are to be copied

Making Image Backups of a Disk

To make an image backup that copies all the files on a disk to a magnetic tape, do the following:

- Use the BACKUP command with the /IMAGE and /RECORD qualifiers as the first part of the command line.
  - /IMAGE qualifier identifies the backup operation as an image backup
  - /RECORD qualifier provides information for BACKUP to use when subsequent backups are taken
- Give the input device followed by a colon
- As the third part of the command line, give the name of the output tape device, followed by a colon and the name of the save set you want to use.

For example, suppose you want to save all the files on a disk named DRA1: to a magnetic tape on the device named MTA0:.

The following command line creates a save set named 19JUNE1991.SAV on the tape that is in MTA0:, and that save set will contain all the files on DRA1:. Before the files are copied, the tape in MTA0: is rewound.

```
$ BACKUP/IMAGE/RECORD  DRA1:   MTA0:19JUNE1991.SAV/REWIND
```

With this command line, you initialize the tape (with the /REWIND qualifier), and the tape has a volume label of 19JUNE (the first six characters of the save set name). With this command, the backup facility also mounts the tape using the save set name that is generated as the label for the tape.

Example 4–1 Image Backup of a Disk

```
$ ALLOCATE MUAO:
$ INITIALIZE MUAO: IMAGE1
$ MOUNT/FOREIGN MUA0:
$ SHOW DEVICE/FILE $1$DUA3:
$ DISMOUNT $1$DUA3:/NOUNLOAD
$ MOUNT $1$DUA3: USERDISK
$ BACKUP/IMAGE/RECORD $1$DUA3: MUA0:PS_15AUG.BCK/REWIND/LABEL=IMAGE1
$ DISMOUNT/NOUNLOAD $1$DUA3:
$ MOUNT/SYSTEM $1$DUA3: USERDISK
```

Example 4–1 shows an example of how you would do an image backup of the disk 1DUA3:.

Notes on Example 4–1:

- IMAGE1 is the label name for MUA0:
- The first DISMOUNT command ensures that there is no disk activity
- USERDISK is the label name for 1DUA3:

Making Incremental Backups of a Disk

Incremental backups save only files that have been created or modified since the last image or incremental backup in which the /RECORD qualifier was used.

To make an incremental backup:

- Use the /SINCE=BACKUP qualifier with the BACKUP command
- The syntax for incremental backups is the same for image backups except that you cannot use the IMAGE qualifier
- You can specify files to be saved, and you can specify input-specifier qualifiers

For example, suppose that you had used the command line shown in Making Image Backups of a Disk for an image backup, and you now wanted to make an incremental backup.

The following command line makes an incremental backup, saving all files on DRA1: that were modified since the previous BACKUP/RECORD command, storing them in a save set named 19JUNE1991.SAV:

```
$ BACKUP/RECORD/SINCE=BACKUP DRA1:[*...]    MTA0:19JUNE1991.SAV/LABEL=219JUNE
```

When you initialize a tape, the tape is given a *volume label*, which is an identifier of one to six characters included as part of the header information on the tape. You can use the /LABEL= qualifier to choose a volume label of up to six characters; alternatively, the first six characters of the save set name are used to form the volume label if the /LABEL qualifier is not used. (For example, if your save set is named 20JUNE1991.SAVE, then the volume label is 20JUNE if you do not use the /LABEL qualifier.) Backup uses the volume label to ensure that you do not create a save set on the wrong magnetic tape, thus unintentionally overwriting existing data.

Example 4–2 shows how you would do an incremental backup of 1DUA3:.

Example 4–2 Incremental Backup of a Disk

```
$ ALLOCATE MUA0:
$ INITIALIZE MUA0: MONDAY
$ MOUNT/FOREIGN MUA0:
$ BACKUP/SINCE=BACKUP/RECORD $1$DUA3:[*...]*.*;* -
MUA0:DUA3_15AUG.BCK/REWIND/LABEL=MONDAY
```

USING COMMAND PROCEDURES FOR BACKUPS

By using command procedures, you can be sure that your system backups take place when you want them to. The following sections give examples of command procedures for specific situations. These are merely examples of the various command procedures that can be used for backup, what you use quite naturally varies according to your users' needs.

Command Procedure for Nightly Image Backups

The following command procedure performs nightly image backups, using disk DUA0:. This procedure executes nightly at 2:00 a.m. This procedure is only useful if your backup does not take more than one tape, otherwise someone needs to be present at 2:00 a.m. to change tapes.

To use the following command procedure:

1. Create the command procedure as shown in the SYS$MANAGER directory ([SYSMGR]), and call it SYSTEM_BACKUP.COM. Edit the command procedure to reflect the name of the disk or disks you want to back up, the name of the tape drive to use, and the name of the save set you want to assign. The example uses a save set named FULL_BACKUP.SAV.
2. Write down the name of the save set that you assigned.
3. Submit the command procedure using the following command line:

   ```
   $ SUBMIT /AFTER="TOMORROW+2:0"  SYS$MANAGER:SYSTEM_BACKUP
   ```

4. Be sure that a tape is loaded on the specified device. When the backup is complete, keep the tape in a safe place and do not use it again until you make another image backup of your system.

```
$!
$! Resubmit this procedure --
$ SUBMIT/AFTER="TOMORROW+2:0" SYS$MANAGER:SYSTEM_BACKUP
$!
$  SET NOON
$  ON ERROR THEN GOTO DONE
$  ON CONTROL_Y THEN GOTO DONE
$  SET PROC/PRIV=ALL
$!
$  REPLY/ALL -
    "Full System Backup About to Begin.  Open Files Will Not Be Saved"
$!
$  BACKUP /IMAGE   DUA0:   MUA0:FULL_BACKUP.SAV /REWIND
$!
$  WRITE SYS$OUTPUT "---> Completed backup of DUA0 save set"
$  WRITE SYS$OUTPUT ""
$!
$DONE:
$  DISMOUNT MUA0:
$  EXIT
```

Command Procedure for Nightly Incremental Backup

The following command procedure performs an incremental backup on three disks every other night at 11:00 p.m.

1. Create the command procedure as shown, and call it INCREMENTAL_BACKUP.COM. Edit the procedure to reflect:

   The names of the disk or disks you are using
   The name of the tape drive to use
   The volume label of the tape
   The name that you want to assign to the save set
   The day of the week (if any) to be omitted in the incremental backup

2. Be sure that an image backup of the system has been made, and also be sure that you continue to make regular image backups of the system.

3. Submit the command procedure using the following command line:

   ```
   $ SUBMIT /AFTER=23  SYS$MANAGER:INCREMENTAL_BACKUP
   ```

4. Be sure that a tape is loaded on the device that you specified. When the incremental backup is complete, keep the tape in a safe place and do not use the tape again until you make another image backup of your system.

```
$!
$! Resubmit this procedure --
$ SUBMIT/AFTER="TOMORROW+23:0" SYS$MANAGER:INCREMENTAL_BACKUP
$!
$ TODAY = f$cvtime("today",,"weekday")
$ IF TODAY .EQS. "Friday" THEN GOTO DONE
$!
$  SET NOON
$  ON ERROR THEN GOTO DONE
$  ON CONTROL_Y THEN GOTO DONE
$  SET PROC/PRIV=(OPER,BYPASS)
$!
$  REPLY/ALL -
    "Incremental Backup About to Begin.  Open Files Will Not Be Saved"
$!
$  BACKUP/RECORD/SINCE=BACKUP  DRA0:,DRA1:,DRA2:  -
   MTA0:INCREMENT.SAV /LABEL=INCREM
$!
$  WRITE SYS$OUTPUT "---> Completed backup of save set"
$  WRITE SYS$OUTPUT ""
$!
$DONE:
$  DISMOUNT MTA0:
$  EXIT
```

RESTORING FILES FROM BACKUP COPIES

You might need to restore files from your backup copies. To restore files, you use the BACKUP command to retrieve the files from a previously created save set.

You can use the BACKUP command to restore the following:

- All the files on a disk (or volume or volume set)
- All the files in a specific directory tree (for example, all the files in a user's main directory and subdirectories)
- One or more specific files
- Files from an incremental backup

The procedure for restoring files depends upon the type of restore operation and whether your most recent backup was an image or incremental backup.

In general, the BACKUP command line that you use to restore files is as follows:

```
BACKUP/qualifiers   save_set_specification   output_specifier
```

Restoring Files from an Image Backup

To restore the entire contents of a disk when your most recent backup was an image backup, use the following procedure:

1. Mount the disk, *to* which you will copy the files, using the MOUNT /FOREIGN command.
2. Load the tape, disk, or diskette that contains the saved backup copy of your disk.
3. Give the BACKUP command with the /IMAGE qualifier, using the following syntax:

```
BACKUP /IMAGE  device:save_set_specification  output_specifier
```

If you do not know the name of the save set, do one of the following:

- If the save set is on a disk, use the DIRECTORY command to determine the name of the save set, for example:

```
$ DIRECTORY BACKUP_DISK:[BACKUPS]
Directory SYS$SYSDEVICE:[BACKUPS]
19APRIL1991.SAV;1
Total of 1 file.
```

The save set is named 19APRIL1991.SAV.

- If the save set is on magnetic tape, load the tape and give the following command, substituting the name of the tape drive you use for *MTA0*:

```
$ BACKUP/LIST/REWIND MTA0:
Listing of save set(s)

Save set:          19APRIL1991.SAV
Written by:        SYSTEM
UIC:               [000001,000004]
Date:              19-APR-1991 22:03:03.63
    .
    .
    .
```

The save set is named 19APRIL1991.SAV.

4. If your backup copy is on more than one tape or diskette, repeat step 2 for each tape or diskette.

5. Dismount the disk onto which you just restored the files, using the /NOUNLOAD qualifier.

The following example shows this process:

```
$ MOUNT/FOREIGN DRA2: ❶
$ BACKUP/IMAGE  MTA1:FULL_BACKUP.SAV/REWIND  DRA2: ❷
$ DISMOUNT/NOUNLOAD  DRA2: ❸
```

In this sequence, the individual command lines do the following:

❶ Logically mount the disk DRA2. The files will be restored to this disk.

❷ Restore the directory structure and all the files from the save set FULL_BACKUP.SAV to the disk DRA2.

The /IMAGE qualifier restores a logical duplicate of the original disk, so that the entire directory structure is restored and the files are placed in the proper directories.

❸ Logically dismount the disk.

Restoring Files from an Incremental Backup

Use the following procedure to restore files after one or more incremental backups:

1. Mount the disk, using DCL MOUNT commands *to* which you will copy the files, using the /FOREIGN qualifier which prepares the file structure of the disk for restoring.
2. Load the tape, disk, or diskette that contains the most recent **image backup** of the disk (or volume or volume set).
3. Give the BACKUP command with the /IMAGE qualifier, using the following syntax:

   ```
   BACKUP/IMAGE device:save_set_specification  output_specifier
   ```

4. If your backup copy is on more than one tape or diskette, repeat step 2 for each tape or diskette.
5. Dismount the disk onto which you have just restored the files from the image backup, using the /NOUNLOAD qualifier.
6. Mount the disk that you are restoring as a file-structured volume, using the following syntax:

   ```
   MOUNT   device_name:  VOLUME LABEL
   ```

7. Dismount the media that contained the image backup, and mount the tape, disk, or diskette that contains the most recent **incremental backup** of the disk (or volume or volume set).
8. Restore your incremental save sets, beginning with the most recent backup. Use the following syntax to restore an incremental backup:

   ```
   BACKUP/INCREMENTAL  save_set_specifier  device_specifier
   ```

   Continue restoring the incremental backups in reverse chronological order, (which allows you to have the most recent changes overwrite previous changes to files which have had activity) until you have processed all of the incremental backups since the most recent image backup. If the incremental backups are on more than one tape or disk, then you must mount each of these successively.

   When you have processed the oldest incremental backup, the restore operation is complete.

The following example shows the process of restoring an entire disk after a series of incremental backups:

```
$ MOUNT/FOREIGN DRA2: ❶
$ BACKUP/IMAGE/RECORD DBA3:WORK_DISK_BACKUP.SAV/SAVE_SET DRA2: ❷
$ DISMOUNT/NOUNLOAD  DRA2: ❸
$ MOUNT DRA2: USER1 ❹
$ BACKUP/INCREMENTAL  DBA3:WORK_DISK_18_JAN.SAV/SAVE_SET  DRA2: ❺
$ BACKUP/INCREMENTAL  DBA3:WORK_DISK_17_JAN.SAV/SAVE_SET  DRA2: ❻
$ BACKUP/INCREMENTAL  DBA3:WORK_DISK_16_JAN.SAV/SAVE_SET  DRA2: ❼
```

In this sequence, the individual command lines do the following:

❶ Logically mount the disk DRA2. The files will be restored to this disk.

❷ Restore the directory structure and all the files from the save set WORK_DISK_BACKUP.SAV to the disk DRA2. This was an image backup, which must be the first save set you restore when you want to restore incremental backup copies.

❸ Logically dismount the disk DRA2.

❹ Remount the disk DRA2, this time as USER1, files-structured volume.

❺ Restore the most recent incremental backup.

❻ Restore the next incremental backup.

❼ Restore the last incremental backup.

Restoring the incremental backups in reverse chronological order is the most efficient way to restore files. When you have restored the last incremental backup, the restoration process is complete.

LISTING THE CONTENTS OF A SAVE SET

To list the contents of a save set, you must use the BACKUP command. There are two ways to list the contents of a save set:

1. Use the /JOURNAL qualifier each time you back up your files. Then, use the BACKUP /JOURNAL/LIST command to list the contents of the save set. You use neither an input specifier nor an output specifier with this command.

2. Make available the save set containing your backup copies by ensuring that the backup media is mounted, and then use the BACKUP/LIST command. With this command, you use the backup media and the save set as your input specifier, and you do not use an output specifier.

For example, suppose you had backed up a disk with this command:

```
$ BACKUP /IMAGE /JOURNAL=SYS$MANAGER:FULL_BACKUP.BJL  -
_$ WORK_DISK:  MTA1:FULL_BACKUP.SAV
```

You can list the contents of the save set on your terminal by using the following command:

```
$ BACKUP /LIST /JOURNAL=SYS$MANAGER:FULL_BACKUP.BJL
```

You can also direct the contents of the save set to be listed in a file that you can read or edit with a text editor by supplying a file specification with the /LIST qualifier.

For example, suppose that the save set containing your backup was on a tape that had been mounted on MTA0. To write the contents of the save set to a text file, enter the following command:

```
$ BACKUP /LIST=SYS$MANAGER:BACKUP_FILES.DAT -
_$ /JOURNAL=SYS$MANAGER:FULL_BACKUP.BJL
```

NOTE

The system manager or operator must keep track of the names of the journal files.

SUMMARY

The BACKUP utility allows you to save copies of all the files on your system.

- There are two types of backups:
  - Image (full) backups
  - Incremental backups
- When you back up files in a save operation, the files are stored in units called save sets.
- The BACKUP command line has three key parts:

```
BACKUP/qualifiers  input_specifier/qualifiers  output_specifier/qualifiers
```

- Command procedures can be used to ensure that system backups take place when you want them to.
- To restore files from your backup copies, use the BACKUP command to retrieve the files from previously created save sets.
- To list the contents of a save set, you must use the BACKUP command and the appropriate qualifiers.

APPENDIX - BACKUP QUICK REFERENCE TABLES

The following tables serve as a quick reference to the various command actions and formats used in the BACKUP utility.

Table 4–1 shows BACKUP command formats for save operations and some of the qualifiers you can use with a save operation.

Table 4–1 Save Operation Quick Reference Table

Command Action	Command Format and Example
Saves a file to a save set on magnetic tape	BACKUP file-spec save-set-specifier/LABEL=label `$ BACKUP STDAT1.DAT MTA0:STDAT1.BCK/LABEL=TAPE01`
Saves the most recent versions of files in a directory to magnetic tape	BACKUP [directory]*.*; save-set-specifier/LABEL=label `$ BACKUP [LYON...]*.*; MTA0:MAR17.BCK/LABEL=W102`
Saves a disk volume to a save set on magnetic tape	BACKUP/IMAGE ddcu: save-set-specifier/LABEL=label `$ BACKUP/IMAGE DBA1: MTA0:9FEB4.BCK/LABEL=M101`

Saves a disk volume to a multivolume save set on more than one magnetic tape drive	BACKUP/IMAGE ddcu: save-set-specifier,ddcu: . . . /LABEL=(label1, . . .)
	`$ BACKUP/IMAGE DBA1: MTA0:17MAR.BCK,MTA1:/ -`
	`_$ LABEL=(WKY101,WKY102)`

Saves a list of files to a save set on magnetic tape	BACKUP file-spec,file-spec,... save-set-specifier/LABEL=label
	`$ BACKUP DBA1:[LYON...]*.PAS,DMA0:[DAKOTA...]*.PAS -`
	`_$ MTA0:PAS17MAR.BCK/LABEL=TAPE01`

Saves a disk volume for incremental backups for the first time	BACKUP/RECORD/IMAGE/LOG ddcu: save-set-specifier/LABEL=label
	`$ BACKUP/RECORD/IMAGE/LOG DBA1: MTA0:9FEB4.BCK/ -`
	`_$ LABEL=DLY101`

Saves a disk volume for incremental backups (not the first time)	BACKUP/RECORD/FAST/LOG ddcu:[*...]/SINCE=BACKUP save-set-specifier/LABEL=label
	`$ BACKUP/RECORD/FAST/LOG DBA1:[*...]/SINCE=BACKUP -`
	`_$ MTA0:928FEB4.BCK/LABEL=DLY101`

Saves an unstructured disk volume	BACKUP/PHYSICAL ddcu: save-set-specifier/LABEL=label `$ BACKUP/PHYSICAL DMA1: MTA0:935FEB4.BCK/LABEL=MTH101`

Table 4–1 Save Operation Quick Reference Table (Cont)

Command Action	Command Format and Example
Saves a directory to a save set on a Files–11 disk	BACKUP [directory] save-set-specifier/SAVE_SET `$ BACKUP [LYON] DBA2:[BACKUP]9FEB3.BCK/SAVE_SET`
Saves a directory to a save set on a Files–11 disk on another node in the network	BACKUP [directory] save-set-specifier/SAVE_SET `$ BACKUP [LYON] CHI10::DBA2:[BACKUP]9FEB3.BCK/SAVE_SET`
Saves a directory tree to a save set on magnetic tape	BACKUP [directory...] save-set-specifier/LABEL=label `$ BACKUP [LYON...] MTA0:1612FEB3.BCK/LABEL=T01`
Saves a directory tree to a save set on magnetic tape and creates a listing file	BACKUP/LIST=file-spec [directory...] save-set-specifier /LABEL=label `$ BACKUP/LIST=8SEP.LOG [LYON...] MTA0:8SEP.BCK /LABEL=WKL101`
Saves a directory tree to a save set on magnetic tape using data compaction to increase the amount of data stored on a tape cartridge	BACKUP [directory...] save-set-specifier /MEDIA_FORMAT=COMPACTION `$ BACKUP [TESTFILES...]*.*;* MUA0:TEST.SAV/MEDIA_FORMAT=COMPACTION/REWIND`

Table 4–2 shows BACKUP command formats for restore operations and some of the qualifiers you can use with restore operations. In the examples in this table, it is assumed that save sets already exist on the magnetic tape and disk.

Table 4–2 Restore Operation Quick Reference Table

Command Action	Command Format and Example

Restores from save set on disk to Files–11 disk with original UICs	BACKUP save-set-specifier/SAVE_SET ddcu:[*...]/BY_OWNER=ORIGINAL
	`$ BACKUP DBA2:[BACKUP]FEB2.BCK/SAVE_SET DBA1:[*...]-`
	`_$/BY_OWNER=ORIGINAL`

Restores from a save set on magnetic tape to a Files–11 disk with original UICs	BACKUP save-set-specifier ddcu:[*...]/BY_OWNER=ORIGINAL `$ BACKUP MTA0:1618FEB2.BCK DBA1:[*...]/BY_OWNER=ORIGINAL`

Restores from save set on disk on another node in the network to Files–11 disk with original UICs	BACKUP save-set-specifier/SAVE_SET ddcu:[*...]/BY_OWNER=ORIGINAL
	`$ BACKUP CHI10::DBA2:[BACKUP]1622FEB2.BCK/SAVE_SET -`
	`_$DBA1:[*...] /BY_OWNER=ORIGINAL`

Restores a selected file in a save set on magnetic tape to a Files–11 disk	BACKUP save-set-specifier/SELECT=file-spec file-spec
	`$ BACKUP MTA0:FEB2.BCK/SELECT=[POUDRE]UPLIFT.PAS -`
	`_$ DBA1:[GEO.PAS]UPLIFT.PAS`

Restores files with a specific UIC to a Files–11 disk	BACKUP save-set-specifier/BY_OWNER=[uic] file-spec
	`$ BACKUP MTA0:1641FEB2.BCK/BY_OWNER=[360,052] -`
	`_$ DBA1:[LYON...]`

Restores files to a Files–11 disk with a new UIC	BACKUP save-set-specifier file-spec/BY_OWNER=[uic]
	`$ BACKUP MTA0:1641FEB2.BCK -`
	`_$ DBA1:[TESTS...]/BY_OWNER=[100,150]`

Restores files to a Files–11 disk; if file exists, creates new version	BACKUP save-set-specifier file-spec/NEW_VERSION `$ BACKUP MTA0:1641FEB2.BCK DBA1:[LYON...]/NEW_VERSION`
Restores files to a Files–11 disk; if file exists, replaces with new version	BACKUP save-set-specifier file-spec/REPLACE `$ BACKUP MTA0:1641FEB2.BCK DBA1:[LYON...]/REPLACE`

Table 4–2 Restore Operation Quick Reference Table (Cont)

Command Action	Command Format and Example

Restores files to a Files–11 disk selecting certain files	BACKUP save-set-specifier/SELECT=file-spec file-spec
	`$ BACKUP MTA0:1641FEB2.BCK/SELECT=[LYON.PAS] -`
	`_$ DBA1:[LYON...]`

Restores a directory tree, placing files in a different subtree	BACKUP save-set-specifier/SELECT=[directory...] [directory2...]
	`$ BACKUP MTA0:1641FEB2.BCK/SELECT=[FIELD...] -`
	`_$ DBA1:[LYON.NEWDATA...]`

Restores a Files–11 volume from a physical save set	BACKUP/PHYSICAL save-set-specifier ddcu: `$ BACKUP/PHYSICAL MTA0:26MAR.BCK DMA3:`
Restores a Files–11 volume from an image save set	BACKUP/IMAGE save-set-specifier ddcu: `$ BACKUP/IMAGE MTA0:17AUG.BCK DRA3:`

Restores a Files–11 volume, maintaining the initialization parameters specified in the DCL command INITIALIZE	INITIALIZE ddcu: volume-name/new-parameters
	MOUNT/FOREIGN ddcu:
	BACKUP/IMAGE save-set-specifier ddcu:/NOINITIALIZE
	`$ INITIALIZE DBA1: UTTLPACK/CLUSTER=5`
	`$ MOUNT/FOREIGN DBA1:`
	`$ BACKUP/IMAGE MTA0:17AUG.BCK DBA1:/NOINITIALIZE`

Table 4–3 shows BACKUP command formats for copy operations, including some of the qualifiers you can use with a copy operation.

Table 4–3 Copy Operation Quick Reference Table

Command Action	Command Format and Example
Copies a directory tree to another directory tree	BACKUP [directory...] [directory...] `$ BACKUP [DAKOTA...] [SUNDANCE...]`
Copies a file to another file	BACKUP file-spec file-spec `$ BACKUP LOGIN.COM [.SAVE]OLDLOGIN.COM`
Copies a disk volume to another disk volume	BACKUP/IMAGE ddcu: ddcu: `$ BACKUP/IMAGE DBA1: DBA2:`
Copies a disk volume to another disk volume using the /PHYSICAL qualifier	BACKUP/PHYSICAL ddcu: ddcu: `$ BACKUP/PHYSICAL DYA1: DYA2:`
Copies two disk volume set using the /IMAGE qualifier	BACKUP/IMAGE volume-set-name ddcu:,ddcu: `$ BACKUP/IMAGE USER$: DBA1:,DBA2:`

Table 4–4 shows BACKUP command formats for compare operations, including some of the qualifiers you can use with a compare operation.

Table 4–4 Compare Operation Quick Reference Table

Command Action	Command Format and Example
Compares two Files–11 files	BACKUP/COMPARE file-spec file-spec `$ BACKUP/COMPARE UPLIFT.EXE;3 UPLIFT.EXE;2`

Compares a selected file from a save set and a Files–11 file	BACKUP/COMPARE save-set-specifier/select=file-spec file-spec
	`$ BACKUP/COMPARE MTA0:FEB2.BCK/SELECT=[POUDRE]UPLIFT.PAS -`
	`_$ UPLIFT.PAS`

Compares an image save set and Files–11 files	BACKUP/COMPARE/IMAGE save-set-specifier ddcu: `$ BACKUP/COMPARE/IMAGE MTA0:12OCT.BCK DRA3:`

Table 4–5 shows BACKUP command formats for a list operation, including some of the qualifiers you can use with a list operation.

Table 4–5 List Operation Quick Reference Table

Command Action	Command Format and Example
Lists the files in a save set at the terminal	BACKUP/LIST save-set-specifier `$ BACKUP/LIST MTA0:1618FEB2.BCK`
Lists the files in a save set, writes to a file	BACKUP/LIST=file-spec save-set-specifier `$ BACKUP/LIST=NEWLIST.LIS MTA0:1618FEB2.BCK`
Lists the files in a save set in full format	BACKUP/LIST/FULL save-set-specifier `$ BACKUP/LIST/FULL MTA0:1618FEB2.BCK`

Lists selected files in a journal file	BACKUP/LIST/JOURNAL=journal-name/selection-qualifiers
	`$ BACKUP/LIST/JOURNAL=SYS$MANAGER:INCBACKUP -`
	`_$ /SELECT=[LYON.WORK...]/SINCE=1-JAN-1991`

Introduction to System Customization

INTRODUCTION

This chapter presents the basic concepts of system customization. Among the topics covered are:

- The definition and identification of the different system startup and login command procedures
- The AUTOGEN and SYSMAN commands used to set system parameters

OBJECTIVES

To describe the task and responsibilities involved with installing and updating system software, a system and network manager should be able to:

- Identify the functions of the different system startup and login command procedures
- Set up a captive account
- Use AUTOGEN and SYSMAN to set system parameters

RESOURCES

- *VMS System Generation Utility Manual*
- *VMS Install Utility Manual*
- *VMS System Manager's Manual*
- *Guide to Setting Up a VMS System*
- *VAX Systems/DECsystems Systems and Options Catalog*
- *VMS I/O User's Reference Manual: Part I*
- *VMS Installation and Operation Guide* for your particular VAX system

TOPICS

- System startup files
  - Site-independent startup file
  - Configuring devices
  - Defining system-wide logical names
  - Executive-mode logical name requirements
  - Installing paging and swapping files
  - General site-specific startup functions
  - Login command procedures
- Maintaining system parameters
  - Utilities for maintaining system parameters
  - Changing physical resources
  - Reconfiguring the system with AUTOGEN
  - Running AUTOGEN
  - SYSMAN parameters
  - Switching window systems

SYSTEM STARTUP FILES

The sequence of operations for STARTUP.COM is as follows:

1. **SYS$MANAGER:SYCONFIG.COM**

   Connects various devices to system and loads their I/O drivers
   Initially empty. If left empty, system automatically configures all devices

2. To add new drivers to the system configuration, the SYSGEN command AUTOCONFIGURE ALL executes unless canceled by the user.

   If the symbol STARTUP$AUTOCONFIGURE_ALL is set to ("0") or "FALSE", this step is not performed.

3. **SWAPFILE1.SYS**, if present, is installed.

4. The CONFIGURE process (swappable) starts. If the SYSGEN parameter NOAUTOCONFIG is set to 1, the CONFIGURE process is not started.

   If the symbol STARTUP$AUTOCONFIGURE_ALL is defined as ("0") or "FALSE" , the step is not performed.

5. **SYS$MANAGER:SYLOGICALS.COM**

   Used to define system-wide logical names

6. **SYS$MANAGER:SATELLITE_PAGE.COM** executes

   Mounts the satellite's local disk
   Installs the paging and swapping files on the satellite's local disk

7. **SYS$MANAGER:SYPAGSWPFILES.COM**

   Used to install paging and swapping files on any disk

8. **SYS$MANAGER:SYSECURITY.COM** executes

   Runs prior to starting the audit server process
   Used to mount or define any disks that hold security auditing log files or local security archive files

9. **SYS$MANAGER:SYSTARTUP_V5.COM**

   General location for site-specific customization commands not addressed by other site-specific startup files
   Overrides commands in STARTUP.COM file

Site-Independent Startup File

SYS$SYSTEM:STARTUP.COM

STARTUP.COM uses a series of component files that accomplish many functions, such as:

- Assigning logical names required by certain VMS system software
- Assigning logical names to the VMS system directories
- Starting up system processes such as
  - JOB_CONTROL
  - QUEUE_MANAGER
  - OPCOM
  - ERRFMT
- Installing known images
- Connecting all standard devices
- Calling the site-specific startup command procedures

These component files are located in a set of directories associated with the system-wide logical name **SYS$STARTUP.**

STARTUP.COM starts up the system:

- Four basic phases (INITIAL, CONFIGURE, DEVICE, BASEENVIRON)
- Three data files (located in SYS$STARTUP)
  - VMS$PHASES.DAT - determines the order of the phases during the startup procedure
  - VMS$VMS.DAT - data file for starting the base VMS operating system environment
  - VMS$LAYERED.DAT - data file for layered products that are installed at system startup

NOTE

Never modify VMS$PHASES.DAT or VMS$VMS.DAT. These files contain important site-independent information that can change from one VMS system release to the next.

Configuring Devices

SYCONFIG.COM

SYS$MANAGER:SYCONFIG.COM is the second site-specific startup file invoked by STARTUP.COM. SYCONFIG.COM is used to connect special devices to the system and load their I/O drivers.

- Only necessary for nonstandard devices or unusual device settings.
- SYSGEN commands are typically placed in this file.
- You can optionally place **MOUNT** commands in this file.
- Most sites mount remaining disks in SYSTARTUP_V5.COM file.
- When SYCONFIG.COM completes, control is returned to STARTUP.COM.
  - STARTUP.COM automatically connects all remaining devices and loads their I/O drivers.
  - Connecting and loading is accomplished by the SYSGEN **AUTOCONFIGURE ALL** command.

Example 5–1 Sample SYCONFIG.COM

```
$!
$! SYS$STARTUP:SYCONFIG.COM
$!
$! Set virtual terminals....
$!
$ RUN SYS$SYSTEM:SYSGEN
CONNECT VTA0/NOADAPTER/DRIVER=TTDRIVER
CONNECT TDA0/NOADAPTER/DRIVER=TDDRIVER
EXIT
$EXIT
```

Defining System-Wide Logical Names

SYLOGICALS.COM

SYS$MANAGER:SYLOGICALS.COM, a site-specific command procedure invoked by STARTUP.COM, is used to define system-wide logical names. A template file is supplied by Digital.

- Define system components as executive-mode logical names.
- To create system logical names:

  ```
  $ ASSIGN/SYSTEM
  $ DEFINE/SYSTEM
  ```

- To delete system logical names:

  ```
  $ DEASSIGN/SYSTEM
  ```

Typically you would use the **/NOLOG** qualifier with either DEFINE or ASSIGN to reduce the amount of printout on the console terminal during system startup.

Table 5–1 lists logical name assignments.

Table 5–2 lists some of the logical names commonly defined in SYLOGICALS.COM for any given VMS operating system site.

Refer to Example 5–3 for a sample SYLOGICALS.COM.

Table 5–1 shows how to assign system logical names.

Table 5–1 Assigning System Logical Names

Operation	Command Format/Example (Requires SYSNAM privilege)

Create or replace a system logical name	`$ ASSIGN/SYSTEM eqv-name log-name`
	`$ DEFINE/SYSTEM log-name eqv-name`
	`$ ASSIGN/SYSTEM SYS$SYSTEM:NOTICE.TXT NOTICE`

Delete a system logical name	`$ DEASSIGN/SYSTEM log-name` `$ DEASSIGN NOTICE`

Table 5–2 lists some standard logical names to define.

Table 5–2 Some Standard Logical Names to Define in SYLOGICALS.COM

Name	Definition	Function
SYS$SYLOGIN	Name of system-wide login command procedure	The system executes this procedure when it creates a process.
SYS$ANNOUNCE	Line of text or name of file containing text	The system displays this line or the contents of the file when the user presses the RETURN key to log in.
SYS$WELCOME	Line of text or name of file containing text	The system displays this line or the contents of the file after a user successfully logs in (by default, "`Welcome to VMS V5.5`")

Executive-Mode Logical Name Requirements

Logical names for some system components and files must be defined in executive-mode.

Examples of components and files you would set up executive-mode logical names for:

- Public disks and directories
- SYSUAF.DAT
- RIGHTSLIST.DAT
- VMSMAIL_PROFILE.DATA
- NETPROXY.DAT

To define an executive-mode logical name:

```
$ DEFINE/SYSTEM/EXECUTIVE/NOLOG  logical-name  equivalence-name
```

Examples of defining logical names in executive mode are shown in Example 5–2.

Example 5–2 Assigning Site-Specific System Logical Names (SYLOGICALS.COM)

```
$!
$!  Assign site-specific logical names
$!
$  ASSIGN /SYSTEM /EXEC /NOLOG  DISK$USER:[PUBLIC]            SYS$PUBLIC
$  ASSIGN /SYSTEM /EXEC /NOLOG  DISK$USER:[TOOLS]             SYS$TOOLS
$  ASSIGN /SYSTEM /EXEC /NOLOG  "This is the MENTOR system"  SYS$ANNOUNCE
$  ASSIGN /SYSTEM /EXEC /NOLOG  "@SYS$MANAGER:WELCOME.TXT"   SYS$WELCOME
$!
```

Example 5–3 Sample SYLOGICALS.COM

```
$ ! SYS$STARTUP:SYLOGICALS.COM
$ !
$ SET NOON
$ !
$ ! Define any site-specific logical names below:
$ !
$ ! This includes site-specific cluster common file definitions (previously
$ ! defined in the site-specific file SYENVIRON.COM).
$ ! The user should include definitions that define the location of
$ ! SYSUAF, NETUAF, VMSMAIL, RIGHTSLIST, NETNODE_REMOTE, and LMF$LICENSE.
$ ! Include a MOUNT/SYSTEM command for the disk that these files reside on.
$ ! See SYS$EXAMPLES:CLU_MOUNT_DISK.COM for the recommended method of doing this.
$ !
$ node = f$getsyi("nodename")
$ !
$ !-------------------------------------------------+
$ ! Define cluster-wide systems operations logicals |
$ !-------------------------------------------------+
$ !
$ ASSIGN/SYSTEM/EXEC WORK4:[COMMON_SYSEXE] SYS$COM_FILES
$ !
$ ASSIGN/SYSTEM/EXEC SYS$COM_FILES:SYSUAF.DAT SYSUAF
$ ASSIGN/SYSTEM/EXEC SYS$COM_FILES:NETPROXY.DAT NETPROXY
$ ASSIGN/SYSTEM/EXEC SYS$COM_FILES:RIGHTSLIST.DAT RIGHTSLIST
$ ASSIGN/SYSTEM/EXEC SYS$COM_FILES:VMSMAIL_PROFILE.DATA  VMSMAIL_PROFILE
$ ASSIGN/SYSTEM/EXEC SYS$COMMON:[SYSEXE]NETNODE_REMOTE.DAT NETNODE_REMOTE
$ !
$ ASSIGN/SYSTEM/EXEC 7 MAIL$SYSTEM_FLAGS
$ ASSIGN/SYSTEM/EXEC SYS$COMMON:[SYSMGR]SYLOGIN SYS$SYLOGIN    !remove .com for MCR CLI
$ ASSIGN/SYSTEM/EXEC "Software Course Development" PSM$ANNOUNCE  ! Header to printouts
$ ASSIGN/SYSTEM/EXEC -
"
  Node: ''node'

    UNAUTHORIZED ACCESS IS PROHIBITED
" -
SYS$ANNOUNCE
$ ASSIGN/SYSTEM/EXEC "@SYS$COMMON:[SYSMGR]WELCOME.TXT"  SYS$WELCOME
$ ASSIGN/SYSTEM 0 SHUTDOWN$MINIMUM_MINUTES
$! Location of COMMON system reports
$ ASSIGN/SYSTEM/EXEC/TRANS=(TERM,CONCEAL) $1$DUA4:[SYS$INFO.] SYS$INFO
$!
$! Define the location of the WORK4:[TEMPLATE] directory
$!
$ DEFINE/EXEC/SYSTEM TEMPLATES   WORK4:[TEMPLATES]
$ DEFINE/EXEC/SYSTEM TEMPLATE    WORK4:[TEMPLATES]
$ !
$ ! Define logical name for DECgraph
$ !
$ ASSIGN/SYSTEM SYS$COMMON:[DECGRAPH] GRAPH$LIBRARY
$ !
$ ! Define logical names specific to Satellite nodes
$ !
$ ! if ("''node'" .EQS. "MOPPET") .OR. ("''node'" .EQS. "BONKRS") THEN $GOTO CI_NODE
$ !
$ !DEFINE /SYSTEM /EXEC DOC$BATCH "''node'_DOC$BATCH"
$ !
$CI_NODE:
$EXIT
```

Installing Paging and Swapping Files

SYPAGSWPFILES.COM

SYS$MANAGER:SYPAGSWPFILES.COM is the third site-specific startup file invoked by STARTUP.COM. It is used to install paging and swapping files on disks other than the system disk.

Before invoking SYPAGSWPFILES.COM, the system activates the following files if they exist in SYS$SYSTEM:

- PAGEFILE.SYS
- SWAPFILE.SYS
- SYSDUMP.DMP

STARTUP.COM then invokes SYPAGSWPFILES.COM.

To activate additional paging and swapping files, insert whatever commands are needed for the installation, including:

- MOUNT
  - Mount disks containing additional swapping and paging files
- SYSGEN
  - Create and/or install additional paging and swapping files

Example 5–4 Sample SYPAGSWPFILES.COM

```
$ SET NOON
$ SET NOCONTROL_Y
$ ! ++
$ ! SYS$MANAGER:SYPAGSWPFILES.COM
$ !
$ ! This is a sample system paging and swapping file command procedure
$ !--
$ MOUNT/SYSTEM $1$DUA4: PGSWP
$ RUN SYS$SYSTEM:SYSGEN
INSTALL $1$DUA4:[SYSEXE]PAGEFILE.SYS/PAGEFILE
INSTALL $1$DUA4:[SYSEXE]SWAPFILE.SYS/SWAPFILE
EXIT
$ EXIT
```

General Site-Specific Startup Functions

SYSTARTUP_V5.COM

SYS$MANAGER:SYSTARTUP_V5.COM is the final site-specific command procedure invoked by STARTUP.COM. A template file is supplied by Digital.

SYSTARTUP_V5.COM is used to accomplish functions not covered by the other site-specific command procedures, such as:

- Mounting public disks
- Setting device characteristics
- Initializing and starting batch and print queues
- Installing known images
- Starting up DECnet software (if it exists)
- Analyzing most recent system failure
- Purging unwanted operator log files
- Starting up the LAT network (if it exists)
- Defining the maximum number of interactive users
- Announcing the VMS system is up
- Starting layered products
- Allowing users to log in

Use separate command procedures for major functions to:

— Keep SYSTARTUP_V5.COM small and manageable

— Allow clean, separate execution of functions after system has been started

  Sometimes needed when certain failures occur and the function needs to be reactivated (For example: print and batch queues)

Example 5–5 illustrates the SYSTARTUP_V5.COM command procedure.

Example 5–5 SYSTARTUP_V5.COM Command Procedure

```
$ SET NOON
$ SET NOCONTROL_Y
$ ! ++
$ ! SYS$MANAGER:SYSTARTUP_V5.COM
$ !
$ ! This is a sample site-specific system startup command procedure
$ !--
$ !
$ ! Create logical name for supporting command procedures
$ DEFINE/NOLOG  STARTUP_PROCS  SYS$SYSROOT:[SYSMGR.STARTUP]
$ !
$ ! Mount site-specific volumes
$ @STARTUP_PROCS:MOUNTDSK.COM
$ !
$ ! Set device characteristics
$ @STARTUP_PROCS:DEVICES.COM
$ !
$ ! Define and start print queues
$ @STARTUP_PROCS:START_PRNT_QUEUE.COM
$ !
$ ! Define and start batch queues
$ @STARTUP_PROCS:START_BATCH_QUEUE.COM
$ !
$ ! Install known images
$ @STARTUP_PROCS:INSTALL.COM
$ !
$ ! Start DECnet software
$ @SYS$MANAGER:STARTNET.COM
$ !
$ ! Start LAT network
$ @SYS$MANAGER:LTLOAD.COM
$ !
$ ! Create reports about the last system failure
$ @STARTUP_PROCS:REPORT_FAILURE.COM
$ !
$ ! Purge old versions of system log files
$ PURGE/KEEP=3 SYS$MANAGER:*.LOG
$ !
$ ! Set the maximum number of interactive users
$ STARTUP$INTERACTIVE_LOGINS == 40
$ !
$ ! Announce availability of the system to all terminals
$ SUBMIT STARTUP_PROCS:START_ANNOUNCE.COM
$ !
$ ! End of SYS$MANAGER:SYSTARTUP_V5.COM
$ !
$ EXIT
```

Login Command Procedures

The VMS operating system provides further capability to specify the user's environment in the form of **login command procedures**.

Login command procedures can be:

- System-wide login command procedures
- Other login command procedures (created by system manager or user)
- Executed each time an interactive process or batch job is created

The VMS system expects to find these command procedures in:

- System logical name SYS$SYLOGIN (system-wide procedure)
- LGICMD field in each user's UAF record
- Default user login file LOGIN.COM

The VMS login command procedure execution sequence:

- If SYS$SYLOGIN is defined, execute the procedure it designates
- If LGICMD contains the name of a command procedure, execute it
- If LGICMD is blank, execute SYS$LOGIN:LOGIN.COM

Common uses of login command procedures include:

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

These command procedures are discussed in Table 5–3.

Example 5–6 shows a sample login to a **captive** account.

Table 5–3 Typical Login Command Procedures (DCL)

Function of Login Command Procedures	Definition of System Logical Name SYS$SYLOGIN and UAF Record Field LGICMD
Personal login only: DCL users create the file LOGIN.COM in their own login default directories to customize their own environments.	SYS$SYLOGIN undefined LGICMD undefined
System and personal login: The system manager creates the command procedure 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$MANAGER:SYLOGIN LGICMD undefined
System and group login: The system manager creates the command procedure SYS$MANAGER:SYLOGIN.COM to customize a common user environment for all system users. Group managers create a command procedure 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 8–3).
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. Typically, 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. Captive accounts may also use the captive login method. Typically, more than one person uses a captive account. When users log in to a captive account, the login procedure runs a program for them, and they communicate with that program. Normally, a user of a captive account never sees the DCL prompt (see Example 5–6).	SYS$SYLOGIN definition optional; affects captive and non-captive users. LGICMD contains the name of a captive command procedure, such as SYS$MANAGER:CAPTIVE, 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 flag. Table 8–12 discusses the FLAGS field of the UAF record further.

Setting Up a Captive Account

Modify the user account to include the CAPTIVE flag and the command procedure name in the LGICMD field. For example:

```
$ RUN AUTHORIZE
UAF> MODIFY SPECULATE /FLAG=CAPTIVE /LGICMD=SYS$MANAGER:CAPTIVE.COM
UAF> EXIT
```

In the above example, SYS$MANAGER:CAPTIVE.COM would call the program for the user. Example 5–7 provides a sample CAPTIVE.COM.

Example 5–6 Using a Captive Account

```
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 04-NOV-1991  16:00:15.16
```

Example 5–7 Captive Command Procedure (CAPTIVE.COM)

```
$! CAPTIVE.COM
$!
$ ON ERROR THEN LOGOUT
$ SET NOCONTROL=Y
$ ON CONTROL-Y THEN LOGOUT
$ WRITE SYS$OUTPUT "Welcome to Speculate"
$ DEFINE/USER SYS$INPUT SYS$COMMAND
$ RUN SYS$SYSTEM:SPECULATE
$ LOGOUT
```

MAINTAINING SYSTEM PARAMETERS

VMS operating system parameters control such things as:

- Sizes of VMS data structures in memory
- Number of installed image files
- Sizes of system files

Automatically customized at system installation:

- SYS$SYSTEM:VAXVMSSYS.PAR (system parameters)
- SYS$MANAGER:VMSIMAGES.DAT (list of images to install)
- Paging file (size)
- Swapping file (size)
- Dump file (size)

Table 5–4 lists these files and their functions.

Table 5–4 System Files

File	Default File Specification	Function
Paging file	SYS$SYSTEM:PAGEFILE.SYS	Manages virtual memory
Swapping file	SYS$SYSTEM:SWAPFILE.SYS	Manages physical memory use
Dump file	SYS$SYSTEM:SYSDUMP.DMP	Saves a partial copy of physical memory when the system fails

Utilities for Maintaining System Parameters

To change current parameter values you need CMKRNL privilege. To make changes to the default parameter values you need write access to the parameter file SYS$SYSTEM:VAXVMSSYS.PAR.

System Generation Utility (SYSGEN)

The system generation utility (SYSGEN) is a system management tool that performs certain privileged system configuration functions:

- Creates and modifies system parameters (not recommended)
- Loads device drivers
- Creates additional paging and swapping files

AUTOGEN Command Procedure

When you boot your system during installation, the AUTOGEN command procedure generates system parameters that are suitable for your hardware configuration. However, if you have an unusual hardware configuration or special workload requirements, you may want to modify some system parameters and rerun AUTOGEN.

NOTE

Digital recommends the use of the AUTOGEN command procedure when modifying system parameters and creating/adding paging and swapping files. For this reason, SYSGEN is not specifically taught in this course.

System Management Utility (SYSMAN)

The system management utility (SYSMAN) centralizes the management of nodes and VAXcluster environments. Rather than logging in to individual nodes and repeating a set of management tasks, SYSMAN allows you to define your management environment to be a:

- Particular node
- Group of nodes
- VAXcluster environment

With a management environment defined, you can perform system management tasks from your local node. SYSMAN executes these tasks on all nodes in the target environment.

Changing Physical Resources

A change in physical resources should be accompanied by a change in:

- System parameter values
- System file sizes

An example of a change in physical resources is when you have:

- Added new hardware
- Added more memory

Tools to use:

- SYS$UPDATE:AUTOGEN.COM - command procedure (recommended method)
  - Determines hardware resources
  - Computes system parameters
  - Creates list of image files to install
  - Calculates size of paging, swapping, and dump files
- SYS$SYSTEM:SYSMAN - utility
  - Changes system parameters
- SYSBOOT - conversational startup utility
  - Changes system parameters

Reconfiguring the System with AUTOGEN

AUTOGEN runs automatically at system installation.

- Determines system hardware resources
- Records appropriate system configuration (to be established at startup)
- Sets system parameter values using SYSGEN utility
- Creates list of images to install
- Creates system files using SYSGEN utility
- Optionally reboots the system to allow new parameters to take effect

You should run AUTOGEN:

- During a new installation or upgrade
- When system physical resources change
- When system workload changes significantly
- When you add a layered (optional) software product. The product installation guide tells you what parameters you need to adjust.
- When you install a shared image
- To specify a system parameter value or file size manually

To run AUTOGEN:

```
$ @SYS$UPDATE:AUTOGEN [start-phase] [end-phase] [execution-mode]
```

Table 5–5 lists AUTOGEN phases. See the *Guide to VMS Performance Management* for more information about AUTOGEN.

Table 5–5 AUTOGEN Phases

Phase	Function	Input Files	Output Files
SAVPARAMS	Records feedback data	None	AGEN$FEEDBACK.DAT
GETDATA	Collects data required for calculations	MODPARAMS.DAT VMSPARAMS.DAT AGEN$FEEDBACK.DAT	PARAMS.DAT
GENPARAMS	Calculates parameter values and file sizes, and generates list of images to install	PARAMS.DAT	SETPARAMS.DAT VMSIMAGES.DAT AGEN$PARAMS.REPORT
TESTFILES	Displays calculated file sizes	PARAMS.DAT	SYS$OUTPUT
GENFILES	Generates new files	PARAMS.DAT	PAGEFILE.SYS SWAPFILE.SYS (and secondary paging and swapping files) SYSDUMP.DMP AGEN$PARAMS.REPORT
SETPARAMS	Saves calculated parameters	SETPARAMS.DAT	VAXVMSSYS.PAR AUTOGEN.PAR VAXVMSSYS.OLD
SHUTDOWN	Shuts down the system	None	None
REBOOT	Reboots to allow new files and parameters to take effect	None	None
HELP	Provides information about AUTOGEN and its phases	None	None

NOTE

All data files are in the directory SYS$SYSTEM:.

Running AUTOGEN

Modifying System Parameters Without Changing File Sizes

1. `$ @SYS$UPDATE:AUTOGEN SAVPARAMS GENPARAMS`
2. Review these files:
   — PARAMS.DAT (information required for AUTOGEN calculations)
   — SETPARAMS.DAT (calculated parameters)
   — AGEN$PARAMS.REPORT (report on feedback data)
3. If you want to change any parameters, edit MODPARAMS.DAT and rerun AUTOGEN as in step 1.
4. `$ @SYS$UPDATE:AUTOGEN SETPARAMS REBOOT`

Changing System Parameters and File Sizes

1. `$ @SYS$UPDATE:AUTOGEN SAVPARAMS TESTFILES`
2. Examine file sizes.
3. If you want to change file sizes, edit MODPARAMS.DAT and rerun AUTOGEN as in step 1.
4. `$ @SYS$UPDATE:AUTOGEN GENPARAMS REBOOT`

Example 5–8 shows a MODPARAMS.DAT file.

Example 5–8 MODPARAMS.DAT File

```
! MODPARAMS.DAT for node DITTO
!
SCSSYSTEMID = 2197
SCSNODE = "DITTO "
PAGEFILE = 60000            !
ADD_GBLPAGES = 425+507+157  ! CMS, BLISS32 and ADA (FJM 9/13/90)
ADD_GBLSECTIONS = 4+5+2     ! CMS, BLISS32 and ADA (FJM 9/13/90)
LOCKIDTBL = 2048            ! FOR RDB (A. B. 9/25/90)
RESHASHTBL = 16384     ! raised for CDD 40+ (D. E. 2/22/91)
MIN_VIRTUALPAGECNT = 136100! for VTX (vhm 5/25/91)
```

ADD →
+=

PuSia

SYSMAN PARAMETERS

The SYSMAN PARAMETERS command lets you perform the following actions on system parameters and parameter files:

- Inspect
- Set
- Modify

The format for the SYSMAN PARAMETERS command is:

```
SYSMAN> PARAMETERS subcommand
```

Table 5–6 SYSMAN PARAMETERS Subcommands

Subcommand	Function
SET	Modifies the value of a system parameter in the work area.
SHOW	Displays the values of system parameters in the work area, plus the default, minimum, and maximum values of the parameters and their units of measure.
USE	Initializes the current work area with system parameter values.
WRITE	Writes the system parameter values to a parameter file, to the current system parameter file, or to the active system in memory.

Example 5–9 SYSMAN PARAMETERS Command

```
                              :

❶ SYSMAN> SET ENVIRONMENT/NODE=BARNUM
  %SYSMAN-I-ENV, current command environment:
   Individual nodes: BARNUM
   Username PRIDE       will be used on nonlocal nodes

❷ SYSMAN> SET PROFILE/PRIVILEGE=CMEXEC

❸ SYSMAN> PARAMETERS SHOW /ACP
❹ %SYSMAN-I-USEACTNOD, a USE ACTIVE has been defaulted on node BARNUM
  Node BARNUM:   Parameters in use: ACTIVE
  Parameter Name             Current    Default    Minimum    Maximum Unit  Dynamic
  --------------             -------    -------    -------    ------- ----  -------
  ACP_MULTIPLE                     0          0          0          1 Boolean     D
  ACP_SHARE                        1          1          0          1 Boolean
  ACP_MAPCACHE                     8          8          1         -1 Pages       D
  ACP_HDRCACHE                    36        128          3         -1 Pages       D
  ACP_DIRCACHE                    36         80          2         -1 Pages       D
  ACP_DINDXCACHE                   9         25          2         -1 Pages       D
  ACP_WORKSET                      0          0          0         -1 Pages       D
  ACP_FIDCACHE                    64         64          0         -1 File-Ids    D
  ACP_EXTCACHE                    64         64          0         -1 Extents     D
  ACP_EXTLIMIT                   100        100          0       1000 Percent/10  D
  ACP_QUOCACHE                    21         64          0         -1 Users       D
  ACP_SYSACC                       4          8          0         -1 Directories D
  ACP_MAXREAD                     32         32          1         64 Blocks      D
  ACP_WINDOW                       7          7          1         -1 Pointers    D
  ACP_WRITEBACK                    1          1          0          1 Boolean     D
  ACP_DATACHECK                    2          2          0          3 Bit-mask    D
  ACP_BASEPRIO                     8          8          4         31 Priority    D
  ACP_SWAPFLGS                    14         15          0         15 Bit-mask    D
  ACP_XQP_RES                      1          1          0          1 Boolean
  ACP_REBLDSYSD                    1          1          0          1 Boolean
```

Notes on Example 5–9:

❶ Establish the target node.

❷ Set the privilege of the profile to enable the next command.

❸ Display the value of a group of parameters. You may display a specific parameter.

❹ Note that USE ACTIVE is assumed.

Switching Window Systems

There may be times that you want to switch windowing systems from VWS to DECwindows or vice versa. To do this:

1. Delete all AUTOGEN FEEDBACK files

```
$ DELETE SYS$SYSTEM:AGEN$*.DAT;*
```

2. Change the the WINDOW_SYSTEM SYSGEN parameter

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> PARAMETERS SET WINDOW_SYSTEM 1  ! or 2 for VWS
SYSMAN> PARAMETERS WRITE CURRENT
SYSMAN> EXIT
```

3. Reboot the system using the following command:

```
$ @SYS$SYSTEM:SHUTDOWN
```

4. After the reboot, execute AUTOGEN using the following command:

```
$ @SYS$UPDATE:AUTOGEN GETDATA REBOOT NOFEEDBACK
```

5. The system will automatically reboot again using the newly generated DECwindows or VWS parameters

SUMMARY

- There is a sequence of operations that STARTUP.COM goes through when starting up the system. STARTUP.COM is also responsible for:
  - Starting up other system startup files that need to be run
  - Executing the AUTOCONFIGURE ALL command unless canceled by the user
  - Installing optional swapping files if they are present
  - Starting the configuration process if the parameter is set accordingly
- After starting up the system, the VMS operating system provides further capability to specify the user's environment using login command procedures. There are two types of login command procedures:
  - System-wide login command procedures
  - Other login command procedures that are created by the system manager or the user
- Using login command procedures you can create a restricted environment for a user to work in.
- Once a system is in operation you can modify system parameters to control such things as:
  - Sizes of VMS data structures in memory
  - Number of installed image files
  - Sizes of system files
- VMS provides three tools to modify these system parameters
  - The SYSGEN utility
  - The AUTOGEN command procedure
  - The SYSMAN utility

Layered Product Installation

INTRODUCTION

This chapter presents the basic concepts used in installing optional (layered) software products. Among the topics covered are:

- The responsibilities of managing product licenses
- The steps used in installing optional (layered) software products

OBJECTIVES

To install optional (layered) software products, a system and network manager should be able to:

- Describe the license management facility (LMF)
- Use the license management facility (LMF) to manage the software license database
- Install optional (layered) software products on a single VMS system or in a VAXcluster environment

RESOURCES

- Installation guide for each product you are installing
- *VMS License Management Utility Manual*

TOPICS

- Overview of optional (layered) software products installation
- The license management facility (LMF)
- Managing product licenses
  - Overview of the license management facility (LMF)
  - Components of the LMF
  - The license management utility (LICENSE)
- Installing layered products
  - Adjusting user privileges and quotas
  - Adjusting SYSGEN parameters
- Installing layered products on a common system disk
- Appendix - License utility subcommands

OVERVIEW OF OPTIONAL (LAYERED) SOFTWARE PRODUCTS INSTALLATION

- Use SYS$UPDATE:VMSLICENSE.COM to register the product authorization key (PAK) for the product.
- Consult optional product's software installation guide for additional steps and further instructions.
  - Some optional software is part of the VMS kit and requires no further installation. These **system integrated products** include:

    DECnet VAX
    VAXcluster software
    VMS Volume Shadowing
    VAX RMS Journaling

  - For most products, you use SYS$UPDATE:VMSINSTAL.COM to copy the product software to the system disk.
- For some products, you must adjust user quotas or system parameters.
- Some products must be installed in a certain order and the product's installation guide should be read carefully to find out what the order is.

THE LICENSE MANAGEMENT FACILITY (LMF)

The following types of software have license keys:

- VMS operating system
  - Separate license for each VMS member
- System integrated products (SIPs)
  - VAXcluster software
  - DECnet VAX software
    - Endnode (DVNETEND)
    - Router (DVNETRTG)
  - VMS Volume Shadowing
- Other layered products

MANAGING PRODUCT LICENSES

Most Digital products now require a **license key** to be installed to operate VMS Version 5.0 and later systems.

Install the key by copying information from a paper product authorization key (PAK). You enter product keys into a cluster-wide database.

Overview of the License Management Facility (LMF)

As each node is started up, licenses are loaded into a volatile LMF database in memory.

- Products on any node in the cluster check the volatile database to determine whether a particular use of the product is licensed or not.
- There are two basic types of licenses:
  - — Availability license: Allows the use of a product on a host
  - — Activity license: Allows a specific number of concurrent users to access a product

Products can also be grouped together under a single license so that a single key will enable the use of several products.

Components of the LMF

Components of the LMF include:

- The LICENSE database
  - Holds all information about keys. The database is managed by the LICENSE utility.
  - SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB is the default location.
  - If there is more than one system disk in the cluster, the logical name LMF$LICENSE should point to a single database for the cluster containing all product licenses for all cluster hosts. This can be defined in SYLOGICAL.COM.
  - If there is no disk accessible to all hosts, separate databases should be maintained **identically**.
- To invoke the license management utility, type:

  ```
  $ LICENSE  subcommand  parameter
  ```

  The license management utility (LICENSE) is a DCL-level interface to the license management facility (LMF) on the VMS operating system.
- SYS$UPDATE:VMSLICENSE.COM
  - A command procedure to assist you in registering your keys in the LICENSE database.

License Units

The following list defines the concept of license unit:

- Element that specifies how much product use a license authorizes.
- Number of individual units purchased with any license is specified by the **license key**.
- Each processor has a series of license unit requirements.
- Products query the LMF to determine if there are sufficient units available to activate the product.

Activity Licenses

An activity license defines the number of concurrent users allowed for a product at one time.

- The user can be anywhere in the cluster.
- Different hosts may require different numbers of units per user.
- As each user activates the product, the number of available units decreases.
- If there are not enough available units, that user gets an error message.
- When a user stops using the product, the units they were using become available.

Table 6–1 shows the values for activity licenses on several different processors.

Table 6–1 Values for an Activity License LURT

VAX Model	Number of License Units Required per User
VAX 8650 system	75
VAX 8350 system	60
VAXstation 2000 system	30

Product Usage with Activity License in a Cluster

As an example, say a product called QUERY has an activity license for 1000 units in a cluster consisting of:

- VAX 8650
- VAX 8350
- Three VAXstation 2000 systems

As each user anywhere in the cluster accesses QUERY, the number of units required per user by that host will be subtracted from the number of units available to all other hosts in the cluster.

- Eight users on the VAX 8650 require 8x75=600 units.
- Five users on the VAX 8350 at the same time require an additional 5x60=300 units.
- This leaves only 100 units.
- Allows each of the three VAXstation 2000s to have one user of QUERY.
- The next user anywhere in the cluster would receive an error message.

The License Management Utility (LICENSE)

LICENSE is a DCL-level interface to the license management facility (LMF) on the VMS operating system. The following format is used to invoke the LMF:

```
$ LICENSE  subcommand  parameter
```

To use the license management utility (LICENSE), enter the LICENSE command and the desired LICENSE subcommand and qualifiers at the DCL prompt ($).

The command procedure **SYS$UPDATE:VMSLICENSE.COM** eliminates much of the typing needed for the LICENSE REGISTER and LICENSE AMEND commands.

Privileges Needed to Use LICENSE:

- Most LICENSE commands require only the privileges needed to access the LICENSE database.
  - Normal VMS file protection applies.
  - The LMF provides the database with a default file access of read and write privileges to system-level processes (S:RW).
- The LICENSE START, LICENSE LOAD, and LICENSE UNLOAD commands need the following privileges:
  - CMKRNL
  - SYSNAM
  - SYSPRV

LICENSE LOAD and LICENSE UNLOAD let you load and unload licenses cluster-wide.

```
SYSMAN> LICENSE LOAD product [/DATABASE=filespec] [/PRODUCER=string]
SYSMAN> LICENSE UNLOAD [product] [/PRODUCER=string]
```

Table 6–2 LICENSE Subcommands

Command	Function
AMEND	Changes a license currently in the LICENSE database
CANCEL	Specifies a new termination date for a product currently in the LICENSE database
CREATE	Creates a LICENSE database with no license records
DISABLE	Disables an existing license in the LICENSE database
ENABLE	Enables an existing license in the LICENSE database so it can be activated with the LICENSE LOAD command
ISSUE	Produces a replica of a PAK that is sent to a file or displayed on your terminal (the default). **This command disables the license in the database.**
LIST	Displays information from the LICENSE database about the specified license or licenses
LOAD	Activates a license or licenses making them available for product authorization for the current node
MODIFY	Modifies a license for system management and license sharing purposes
REGISTER	Adds a new license to the LICENSE database
START	Sets up an in-memory table for your system, and activates all licenses that are registered and enabled in the LICENSE database
UNLOAD	Deactivates a license, making the product unavailable from the current node

Example 6–1 shows a sample product authorization key.

Example 6–1 Product Authorization Key

```
 - - - - - - -
| | | | | | | |          LICENSE SOFTWARE PRODUCT          | DOCUMENT ISSUE DATE |
|d|i|g|i|t|a|l|          PRODUCT AUTHORIZATION KEY         |     09-MAY-1991     |
| | | | | | | |                                            |_____________________|
 - - - - - - -
Digital Equipment Corporation
Maynard, MA.

| LICENSE ADMINISTRATION LOCATION:   |  ORDERED BY: Newton Scientific Inst.
|                                    |              Mr. Isaac Newton
| Digital Equipment Corporation      |              128 Main St.
| Maynard, Massachusetts             |              Newton, MA 03300
|                                    |
|                                    |
|                                    |
 ------------------------------------
********************************************************************************
 PAK ID:
                   Issuer: DEC
     Authorization Number: USA000877

 PRODUCT ID:
             Product Name: FORTRAN
                 Producer: DEC

 NUMBER OF UNITS:
          Number of units: 5000

 KEY LEVEL:
                  Version: 5.4
     Product Release Date: 18-APR-1991

 KEY TERMINATION DATE:
     Key Termination Date:

 RATING:
  Availability Table Code: F
      Activity Table Code:

 MISCELLANEOUS:
              Key Options: MOD_UNITS
            Product Token:
              Hardware-Id:
                 Checksum: 1-CCLB-MNBO-KNNG-CBEH
********************************************************************************
```

Example 6–2 shows a sample VMSLICENSE session.

Example 6–2 VMSLICENSE Session

```
$ SET DEFAULT SYS$UPDATE
$ @VMSLICENSE.COM

    VMS License Management Utility Options:

        1. Register a Product Authorization Key
        2. Amend an existing Product Authorization Key
        3. Cancel an existing Product Authorization Key
        4. List Product Authorization Keys
        5. Modify an existing Product Authorization Key

        9. Exit this procedure

    Type '?' at any prompt for a description of the information
    requested.

Enter one of the above choices [1]: 1
Do you have your Product Authorization Key? [YES]: y

   The REGISTER option allows you to add a new license to a license
   database.  A Product Authorization Key (PAK) provides the product
   name and information you need to register the license.  You must
   enter all the information provided by your PAK exactly as specified.

PAK ID:
                      Issuer [DEC]:
        Authorization Number []: USA000877

PRODUCT ID:
                Product Name []: FORTRAN
                    Producer [DEC]:

NUMBER OF UNITS:
             Number of Units []: 5000

KEY LEVEL:
                     Version []: 5.4
        Product Release Date []: 18-APR-1990

KEY TERMINATION DATE:
        Key Termination Date []:

RATING:
     Availability Table Code []: F
         Activity Table Code []:
```

Example 6–2 VMSLICENSE Session (Cont.)

```
                              :
MISCELLANEOUS:
                  Key Options []: MOD_UNITS
                Product Token []:
                  Hardware-Id []:
                     Checksum []: 1-CCLB-MNBO-KNNG-CBEH

      License Database File:  SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB
                     Issuer:  DEC
              Authorization:  USA000877
                   Producer:  DEC
               Product Name:  FORTRAN
                      Units:  5000
                       Date:  18-APR-1990
                    Version:  5.4
           Termination Date:
               Availability:  F
                   Activity:
                    Options:  MOD_UNITS
                      Token:
                Hardware ID:
                   Checksum:  1-CCLB-MNBO-KNNG-CBEH
Is this information correct? [YES]:
Registering FORTRAN license in SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB...
Do you want to LOAD this license on this system? [YES]:
%LICENSE-I-LOADED, DEC FORTRAN was successfully loaded with 5000 units
    VMS License Management Utility Options:
        1. Register a Product Authorization Key
        .
        .  <other options omitted from this listing>
        .
        9. Exit this procedure
Enter one of the above choices [1]: 9
$
```

LICENSE MANAGEMENT FACILITY AND LICENSE AGREEMENTS

The terms and conditions of your product contract determine your legal use of software. The LMF is a management tool that can help you comply with your license agreement. However, the LMF offers options for many kinds of license agreements. Using some of these options can be illegal for your specific contract. You must read your contract carefully to determine which LMF options you can use legally.

For more information on your legal responsibilities, contact your Digital representatives.

Adjusting User Privileges and Quotas

For a user to use a layered product, you may need to modify their user account privileges or quotas. User account privileges and quotas are stored in the file SYSUAF.DAT. You use the Authorize utility to verify and change user account privileges and quotas. To use the Authorize utility, set your default directory to SYS$SYSTEM and enter the DCL RUN command as follows:

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

At the Authorize utility prompt (UAF>), to check a particular account, enter the SHOW command with an account name. For example:

```
UAF> SHOW SMITH
```

To change a privilege or quota, enter the MODIFY command in the following format:

MODIFY account-name/PRIVILEGE=privilege-name/quota-name=nnnn

The following example adds the NETMBX privilege, modifies the BYTLIM quota for the SMITH account, and exits the utility:

```
UAF> MODIFY SMITH /PRIVILEGE=NETMBX /BYTLIM=16384
UAF> EXIT
```

After you exit the utility, the VMS operating system displays messages indicating whether or not changes were made. Once you have made the changes, users must log out and log in again for the new privileges and quotas to take effect.

Whenever a new account is created, its characteristics come from the DEFAULT account except for the ones that you specify. To change a limit or quota for any future accounts, modify them in the DEFAULT account that exists in every SYSUAF.DAT file.

Table 6–3 AUTHORIZE Qualifiers for Quota Fields

Qualifier	Function
/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 one time for the user's process
/PRIORITY=value	The default base priority for all processes created by the user
/SHRFILLM=value	Maximum number of shared files the user may 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

Adjusting SYSGEN Parameters

Some installation guides may specify that you must ensure that you have a certain number of GBLPAGES (global pages) and GBLSECTIONS (global sections) on your system. You do this as follows:

1. Invoke the Install utility by typing

   ```
   $ INSTALL
   ```

2. Display the existing global sections already known to VMS by typing

   ```
   INSTALL> LIST /GLOBAL/SUMMARY
   ```

   The Install utility then lists the following:

   - The number of global sections used
   - The number of global pages used and unused

Type CTRL/Z to exit the Install utility

Example 6–3 illustrates the output from the LIST/GLOBAL/SUMMARY command.

Example 6–3 Displaying the Existing Global Sections

```
$ INSTALL
INSTALL> LIST/GLOBAL/SUMMARY

  Summary of Local Memory Global Sections
    279 Global Sections Used,  23722/278 Global Pages Used/Unused

INSTALL> ^Z
```

3. Display the existing number of available global sections and global pages by typing:

   ```
   $ RUN SYS$SYSTEM:SYSGEN
   SYSGEN> USE CURRENT
   SYSGEN> SHOW GBLSECTIONS
   SYSGEN> SHOW GBLPAGES
   SYSGEN> EXIT
   ```

   The system responds with data on global sections and global pages. The current maximum number of global sections and pages is the first number given in the respective system response.

Example 6–4 illustrates the output from the SYSGEN> SHOW GBLSECTIONS and GBLPAGES commands.

Example 6–4 Displaying the Available Global Sections and Pages

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> USE CURRENT
SYSGEN> SHOW GBLSECTIONS

Parameter Name          Current    Default     Min.     Max.     Unit  Dynamic
--------------          -------    -------   -------  -------    ----  -------
GBLSECTIONS                 340        250       60      4095 Sections

SYSGEN> SHOW GBLPAGES

Parameter Name          Current    Default     Min.     Max.     Unit  Dynamic
--------------          -------    -------   -------  -------    ----  -------
GBLPAGES                  24000      10000      512        -1 Pages

SYSGEN> EXIT
```

Compare these numbers with the numbers currently used, as shown in step 2 of the Install utility. If fewer than the required number of global pages or global sections remain, you should increase the number of available global sections or pages as follows:

- Edit the file SYS$SYSTEM:MODPARAMS.DAT by adding or modifying the applicable lines indicated below (n and m are, respectively, the total number of global sections and global pages required for the new images added):

  ADD-GBLSECTIONS=n
  ADD-GBLPAGES=m

- Invoke the AUTOGEN utility to update the GBLSECTIONS and GBLPAGES parameters:

  ```
  $ @SYS$UPDATE:AUTOGEN SAVPARAMS REBOOT
  ```

Table 6–4 Commonly Set SYSGEN Parameters

Parameter	Function
GBLPAGES	Establishes the size of the global page table and the limit for the total number of global pages that can be created
GBLSECTIONS	Determines the maximum number of global sections that can be made known to the system by allocating the necessary storage for the GST entries
VIRTUALPAGECNT	Determines the total number of pages that can be mapped for a process, which can be divided in any fashion between P0 and P1 space

Example 6–5 illustrates how to install VAX FORTRAN using the VMSINSTAL command procedure.

Example 6–5 Using the VMSINSTAL Command Procedure to Install VAX FORTRAN

```
$ @SYS$UPDATE:VMSINSTAL FORT050 MUA0:

        VAX/VMS Software Product Installation Procedure V5.4

It is 14-OCT-1991 at 12:11.
Enter a question mark (?) at any time for help.

* Are you satisfied with the backup of your system disk [YES]? RETURN

The following products will be processed:
  FORT V5.4

        Beginning installation of FORT V5.4 at 12:12

%VMSINSTAL-I-RESTORE, Restoring product saveset A ...
%VMSINSTAL-I-RELMOVED , The product's release notes have been successfully
moved

        Product:      FORTRAN
        Producer:     DEC
        Version:      5.4
        Release Date: 1-FEB-1990

* Does this product have an authorization key registered and loaded? YES

* Do you want to purge files replaced by this installation [YES]? RETURN

* Do you want to install the VAX FORTRAN compiler [YES]? RETURN

        This kit contains an Installation Verification Procedure
        (IVP) to  verify  the  correct  installation of the  VAX
        FORTRAN compiler.  The IVP will be left in:
             SYS$SYSROOT:[SYSTEST.FORTRAN]FORTRAN$IVP.COM.
        After the  installation is  complete, you can invoke the
        command file at any time to reverify that VAX FORTRAN is
        installed and working correctly.
```

Example 6–5 Using the VMSINSTAL Command Procedure to Install VAX FORTRAN (Cont.)

```
* Do you want to run the IVP after the installation [YES]? [RETURN]

        This kit  contains a file  summarizing the new features,
        changes, restrictions, and  compatibility issues in this
        release of  VAX  FORTRAN.   The  name of  this  file  is
        FORT050.RELEASE_NOTES and it is placed in SYS$HELP:.

        This file  contains  information valuable to VAX FORTRAN
        programmers.  Please inform your  user community of this
        file's existence.

        This kit also  contains the  file,  FORTRANFIXES050.MEM,
        summarizing  the  bug  fixes  made to  the  VAX  FORTRAN
        compiler  since its  last  release.  This  file  will be
        placed in SYS$HELP:.

* Would you like a copy of it printed now? [NO]? [RETURN]

        In order to build your FORSYSDEF library, this procedure
        requires at least  6000  blocks of available disk space,
        most of  which is used  for  temporary work files.   The
        FORSYSDEF  library  itself will  take approximately 1900
        blocks of disk space upon completion  of  this procedure
        and will be placed in your SYS$LIBRARY area.

                                 NOTE

        Before installing FORSYSDEF,   be sure to  have read the
        appropriate  section  of  the installation  guide  which
        addresses the question  of when a  new FORSYSDEF  should
        be built.

* Do you want to build a new FORSYSDEF.TLB [NO]? YES

* Do you want to install FORTRAN help [YES]? [RETURN]

        This  kit contains  two  separate  HELP files,  a  large
        version (approximately 600 blocks) including information
        on   FORTRAN  language features,  and  a smaller version
        (approximately 100 blocks)  describing  only the FORTRAN
        command.
```

Example 6–5 Using the VMSINSTAL Command Procedure to Install VAX FORTRAN (Cont.)

```
* Do you want to install the larger version of FORTRAN help [YES]? [RETURN]

        All questions  regarding the installation of VAX FORTRAN
        have now been asked.  Depending upon your configuration,
        time estimates for the installation(s) have been provided.

        VAX FORTRAN compiler:   3  to  60 minutes
        FORSYSDEF.TLB:          10 to 120 minutes
        FORTRAN HELP:           1  to  15 minutes

%VMSINSTAL-I-SYSDIR, This product creates system disk directory  VMI$ROOT:[SYST
EST.FORTRAN
%CREATE-I-EXISTS, VMI$ROOT:[SYSTEST.FORTRAN] already exists

        +--------------------------------------------------------+
        |       Installing the VAX FORTRAN V5 Compiler           |
        +--------------------------------------------------------+

        +--------------------------------------------------------+
        |               Installing FORSYSDEF.TLB                 |
        +--------------------------------------------------------+

        +--------------------------------------------------------+
        |              Installing VAX FORTRAN HELP               |
        +--------------------------------------------------------+

        Your  VMS  system  will now be updated  to  include  the
        following new and modified file(s):

        SYS$HELP:FORT050.RELEASE_NOTES  [new]
        SYS$SYSTEM:FORTRAN.EXE          [new]
        SYS$MESSAGE:FORTERR1.EXE        [new]
        SYS$MESSAGE:FORTERR2.EXE        [new]
        SYS$LIBRARY:FORTV5CLD.CLD       [new]
        SYS$LIBRARY:DCLTABLES.EXE       [modified]
        SYS$HELP:FORTRANFIXES050.MEM    [new]
        SYS$TEST:FORTRAN$IVP.COM        [new]
        SYS$LIBRARY:FORSYSDEF.TLB       [new]
        SYS$TEST:FORSYSDEFTST.COM       [new]
        SYS$HELP:HELPLIB.HLB            [modified]
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target directories...

        +--------------------------------------------------------+
        |           Verification Command Procedure for           |
        |                      VAX FORTRAN                       |
        +--------------------------------------------------------+

VAX FORTRAN V5.4-34 TEST PASSED

        VMSINSTAL procedure done at 12:36
```

Suppose you would like to copy the layered product kit to a disk for later installation.

Example 6–6 illustrates how you would use VMSINSTAL to transfer a layered product kit from a magnetic tape to a disk.

Example 6–6 Using the VMSINSTAL Command Procedure to Make a Copy of a Layered Product Kit

```
$ @SYS$UPDATE:VMSINSTAL * MUB0: OPTIONS G $1$DUA1:[MOPPET] -
_$ "/VERIFY/LOG/CONFIRM"

 VAX/VMS Software Product Installation Procedure V5.4-1P

It is 13-JUN-1991 at 09:09.

Enter a question mark (?) at any time for help.

Please mount the first volume of the set on  MUB0:.
* Are you ready? y

%MOUNT-I-MOUNTED, MES023 mounted on _BROWNY$MUB0:

The following products will be processed:
  MESS V2.3

%VMSINSTAL-I-CREATEDIR, Creating temporary directory  _$1$DUA1:[MESWORK].

    Because VMSINSTAL does not know how many save sets comprise a software
    product, it will simply copy as many as it can find.  Do not be
    concerned about error messages from BACKUP after all save sets have
    been copied.

 Getting save sets for MESS V2.3
%VMSINSTAL-I-RESTORE, Restoring product save set A ...
_$1$DUA1:[MESWORK]KITINSTAL.COM;12, copy? (Y or N): y
_$1$DUA1:[MESWORK]MESSAGE.CLD;1, copy? (Y or N): y
_$1$DUA1:[MESWORK]MESSAGE_INSTAL.COM;11, copy? (Y or N): y
_$1$DUA1:[MESWORK]MESSAGE_SETUP.COM;2, copy? (Y or N): y
%BACKUP-S-COPIED, copied _$1$DUA1:[MESWORK]KITINSTAL.COM;12
%BACKUP-S-COPIED, copied _$1$DUA1:[MESWORK]MESSAGE.CLD;1
%BACKUP-S-COPIED, copied _$1$DUA1:[MESWORK]MESSAGE_INSTAL.COM;11
%BACKUP-S-COPIED, copied _$1$DUA1:[MESWORK]MESSAGE_SETUP.COM;2
%BACKUP-I-STARTVERIFY, starting verification pass
%BACKUP-S-COMPARED, compared _$1$DUA1:[MESWORK]KITINSTAL.COM;12
%BACKUP-S-COMPARED, compared _$1$DUA1:[MESWORK]MESSAGE.CLD;1
%BACKUP-S-COMPARED, compared _$1$DUA1:[MESWORK]MESSAGE_INSTAL.COM;11
%BACKUP-S-COMPARED, compared _$1$DUA1:[MESWORK]MESSAGE_SETUP.COM;2
%VMSINSTAL-I-RESTORE, Restoring product save set B ...
_$1$DUA1:[MESWORK]MESSAGE.OPT;8, copy? (Y or N): y
_$1$DUA1:[MESWORK]MESSAGE.OBJ;1, copy? (Y or N): y
_$1$DUA1:[MESWORK]VAX_MESSAGE.OLB;21, copy? (Y or N): y
%BACKUP-S-COPIED, copied _$1$DUA1:[MESWORK]MESSAGE.OPT;8
%BACKUP-S-COPIED, copied _$1$DUA1:[MESWORK]MESSAGE.OBJ;1
%BACKUP-S-COPIED, copied _$1$DUA1:[MESWORK]VAX_MESSAGE.OLB;21
%BACKUP-I-STARTVERIFY, starting verification pass
%BACKUP-S-COMPARED, compared _$1$DUA1:[MESWORK]MESSAGE.OPT;8
%BACKUP-S-COMPARED, compared _$1$DUA1:[MESWORK]MESSAGE.OBJ;1
%BACKUP-S-COMPARED, compared _$1$DUA1:[MESWORK]VAX_MESSAGE.OLB;21
```

Example 6–7 Using the VMSINSTAL Command Procedure to Make a Copy of a Layered Product Kit (Cont.)

```
%VMSINSTAL-I-RESTORE, Restoring product save set C ...

%BACKUP-F-OPENIN, error opening MUB0:[000000]MES023.C; as input
-SYSTEM-W-NOSUCHFILE, no such file
%VMSINSTAL-E-NOSAVESET, Save set  C  cannot be restored.

 A total of 2  save sets copied for  MESS V2.3

 VMSINSTAL procedure done at 09:15
$
$ DIRECTORY
Directory $1$DUA1:[MOPPET]

MES023.A;1          MES023.B;1          SUMMARY.LOG;1

Total of 3 files.
```

VMSINSTAL Installer's Options Overview

Table 6–5 VMSINSTAL Options

Option	Function
A	The *auto-answer* option makes it easier to reinstall a product by providing responses to VMSINSTAL questions and prompts during the reinstallation. The auto-answer option is used most often for reinstalling products after a system is upgraded.
AWD	The *alternate working device* option lets you specify an alternate working device for the temporary working directory. This option enables you to perform an installation with fewer free blocks on the VMI$ROOT device than is otherwise required.
G	The *GET save set* option is used to copy the product kit save sets into a disk directory or other storage device for later installation. When option G is selected, all kit save sets are copied, but no installation is performed.
L	The *file log* option is used to log all file activity to the controlling terminal during installation. File activity is defined as any action that alters the disposition of a file, such as creating a new file, updating a library, or deleting a file.
N	The *release notes* option is used to display or print the release notes file supplied by the layered product.
R	The *alternate root* option is used to install the product in a system root other than that of the running system. This option makes it possible to test a new product without disturbing the running system.

INSTALLING LAYERED PRODUCTS ON A COMMON SYSTEM DISK

The standard precautions for product installation hold also for VAXcluster systems. For most products, the only concerns that may require more careful attention are SYSGEN parameters and licensing.

- Install layered products as in a nonclustered environment.
- Perform the actual installation (the documented procedure) once for each system disk.
- Edit MODPARAMS.DAT to add required number of global pages and sections. Either edit the cluster common MODPARAMS.DAT file or the MODPARAMS.DAT file on each node in the cluster.

After installation:

- Create product-specific files in the SYS$SPECIFIC directory on each node, if necessary.
  - VMSINSTAL will tell you if you have to create a directory in SYS$SPECIFIC.
- Modify any files in SYS$SPECIFIC on each node that the procedure told you to modify.
- Reboot each node to ensure that:
  - The node is set up to run the product correctly
  - The node is running the latest version of the product
- Manually run the installation verification procedure (IVP), if you did not run it during the installation procedure.
  - Run it from at least one other node in the cluster, preferably from all nodes.
  - If VMSINSTAL deletes the IVP, you may be able to restore it with the following command procedure:

```
$ @VMSINSTAL  product  source_device  OPTIONS G  device:[directory]
$ BACKUP  device:[directory]product.A/SELECT=KITINSTAL.COM  device:[directory]
$ @device:[directory]:KITINSTAL  VMI_IVP
```

Do not use search lists when creating files:

- Use SYS$SPECIFIC or SYS$COMMON instead of SYS$SYSROOT
- Use SYS$SPECIFIC:[SYSEXE] or SYS$COMMON:[SYSEXE]) instead of SYS$SYSTEM

SUMMARY

- Once the VMS operating system is installed, you can install other optional (layered) software products. VMS provides a tool called VMSINSTAL to help you install most layered products.
- To run a layered product you must purchase a license which comes in the form of a product authorization key (PAK). It is recommended that you install the PAK using the license management facility (LMF) before installing the product.
- You may need to modify a user's quotas or resource limits to enable them to use the layered product.
- System parameters may need to be adjusted to install the layered product.

APPENDIX - LICENSE UTILITY SUBCOMMANDS

AMEND

- Amends a license currently in the LICENSE database.
- Use the LICENSE AMEND command only when the software vendor provides amendment information. (Currently, Digital does not issue amendments to licenses.)
- Use the LICENSE MODIFY command for all other changes to a license.

Format: LICENSE AMEND product-name

Example:

```
$ LICENSE AMEND GIZMO /PRODUCER=DEC /ISSUER=DEC -
_$ /AUTHORIZATION=USA4321 -
_$ /CHECKSUM=1-GEAD-OODA-HIDN-PLAC /VERSION=9.3
```

This command amends the license for the Digital software product named GIZMO. Entering this command upgrades an existing GIZMO license to Version 9.3. The producer name, issuer name, authorization number, and checksum number are typed exactly as they appear in the amendment information.

CANCEL

Specifies a new termination date for a product currently in the LICENSE database.
You must use the /TERMINATION=date qualifier.

Format: LICENSE CANCEL /TERMINATION=date product-name

Example:

```
$ LICENSE CANCEL/AUTHORIZATION=USA1776 -
_$ /TERMINATION=04-JUL-1990 VAX-VMS
```

Unless an earlier termination date exists, this command sets a new cancellation date of July 4, 1990 for the license on the VMS system.

Note that the product name is entered with a hyphen (-) character as it was specified on the PAK.

COPY

Copies one or more licenses from one LICENSE database to another. When you use the LICENSE COPY command, LMF disables the source license and registers a copy in the destination license database as if it were a new license. If the terms and conditions of your license contract allow it, you can reenable the source database license by using the LICENSE ENABLE command.

The LICENSE COPY command cannot be used to create a copy of a license in the same database as the source of the copy.

Format: LICENSE COPY product-name[,...] output-database

Example:

```
$ LICENSE COPY FORTRAN BACKUP_DATA:BACKUP.LDB
```

CREATE

Creates a LICENSE database with no license records. LMF creates a default LICENSE database in SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB. Therefore, you need not specify this command.

Format: LICENSE CREATE

Example:

```
$ LICENSE CREATE/DATABASE=SYS$MANAGER:LMF$LICENSE.LDB
```

DELETE

Deletes one or more licenses and all history information for those licenses from the LICENSE database. Deleted licenses are no longer available to the system for any use.

Format: LICENSE DELETE product-name

Example:

```
$ LICENSE DELETE FORTRAN, COBOL, PASCAL
```

DISABLE

Disables an existing license in the LICENSE database. A disabled license cannot be activated to authorize product use. The LICENSE DISABLE command does not immediately affect any active processes. Active processes are affected only if you enter a LICENSE UNLOAD command or if the VMS system shuts down.

Format: LICENSE DISABLE product-name

Example:

```
$ LICENSE DISABLE VAXset /PRODUCER=DEC
```

This command disables the license for VAXset software, produced by Digital. Because no database is specified, LMF uses the default database.

ENABLE

Enables an existing license in the LICENSE database so that it can be activated with the LICENSE LOAD command. This command cancels the effect of the LICENSE DISABLE command. Newly registered licenses are enabled by default.

Format: LICENSE ENABLE product-name

Example:

```
$  LICENSE ENABLE VAXSET /PRODUCER=DEC
```

ISSUE

Produces a replica of a PAK that is sent to a file or displayed on your terminal (the default). If the terms and conditions of your license contract allow it, you can then enter this PAK replica in the LICENSE database of another processor. When you enter a LICENSE ISSUE command, LMF disables the license in the current LICENSE database and marks the license ISSUED. To enable a license that has been marked ISSUED, enter the LICENSE ENABLE command.

Format: LICENSE ISSUE product-name

Example:

```
$  LICENSE ISSUE /OUTPUT=SYS$MANAGER:FORTRAN.PAK /PRODUCER=DEC FORTRAN
```

LIST

Displays information from the LICENSE database about the specified license or licenses. You can control the form, content, and location of information displayed with the /BRIEF, /FULL, /HISTORY, and /OUTPUT qualifiers.

Format: LICENSE LIST [product-name]

Example

```
$  LICENSE LIST
Press CTRL/Z to exit, use arrow keys to scroll.
.
.
.
-----------------------------------

 FORTRAN                    DEC
 COBOL                      DEC
 PASCAL                     DEC
[End of List]
```

History records are written by every command that changes any fields in a license record. These commands are AMEND, CANCEL, ENABLE, DISABLE, ISSUE, and MODIFY.

Example 6–8 LICENSE LIST/FULL/HISTORY Output

```
  $  LICENSE LIST /FULL /HISTORY FORTRAN
Press CTRL/Z to exit, PF3-PF4 for Previous-Next Screen and Arrow Keys to Scroll.

   License Management Facility

   LICENSE database File:       ART::SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB
   Created on:                  17-JUN-1990
   Created by user:             MONET
   LMF Version:                 V1.0

   ------------------------------------------------------
   Issuer:                      DEC
   Authorization:               USA-2468
   Product Name:                FORTRAN
   Producer:                    DEC
   Units:                       2000
   Version:                     V4.7
   Date:                        (none)
   Termination Date:            10-DEC-1990
   Availability:                F (Layered Products)
   Activity:                    0
   Options:
   Hardware ID:

   Revision Level:              2
   Status:                      Active
   Command:                     AMEND
   Modified by user:            DEGAS
   Modified on:                 19-JUN-1990 14:32:23.41
   Include:                     ART

   -----------------------------------
   Issuer:                      DEC
   Authorization:               USA-2468
   Product Name:                FORTRAN
   Producer:                    DEC
   Units:                       2000
   Modified Units:              9999
   Date:                        (none)
   Version:                     V4.5
   Termination Date:            20-JUN-1990
   Availability:                F (Layered Products)
   Activity:                    0
   Options:                     MOD_UNITS
   Hardware ID:

   Revision Level:              1
   Status:                      History
   Command:                     AMEND
   Modified by user:            DEGAS
   Modified on:                 29-JUN-1990 12:12:27.33
  [End of List]
```

LOAD

Activates a license or licenses making them available for product authorization for the current node. The product license or licenses must currently exist and be active in the LICENSE database. If the license is already loaded, the LMF returns an error message and makes no changes.

To use this command you need the privilege to change mode to kernel (CMKRNL), the privilege to create system logical names (SYSNAM), and the system privilege (SYSPRV).

Format: LICENSE LOAD [product-name]

Example:

```
$  LICENSE UNLOAD FORTRAN
$  LICENSE MODIFY/INCLUDE=MUSIC FORTRAN
$  LICENSE LOAD FORTRAN
```

Whenever a load is successful, the LICENSE utility displays a message showing the number of license units loaded. You can also use the DCL command SHOW LICENSE to see what licenses are loaded.

MODIFY

Modifies a license for system management and license-sharing purposes. The LICENSE MODIFY command changes data in the LICENSE database immediately, but the modifications do not affect a running system until you activate the modified license with a LICENSE LOAD command.

Before using this command, refer to your software license agreement to determine whether the modifications you want to make are valid under the terms of the license.

Format: LICENSE MODIFY product-name

Example:

```
$  LICENSE MODIFY /EXCLUDE=(DANCE,THEATR) -
_$  /COMMENT="Modified to exclude nodes DANCE & THEATR 6/23/90" -
_$   FORTRAN
```

MOVE

Moves one or more licenses from one LICENSE database to another. When you use the LICENSE MOVE command, LMF deletes the licenses in the source LICENSE database.

Format: LICENSE MOVE product-name[,..] output-database

Example:

```
$  LICENSE MOVE FORTRAN ALT_SYS2:LMF$LICENSE.LDB
```

REGISTER

Adds a new license to the LICENSE database. A product authorization key (PAK) provides the product name and information you need to register the license. You must enter all information provided by your PAK exactly as specified.

Often the command procedure SYS$UPDATE:VMSLICENSE.COM is used to register a new product license. This provides a prompt-based interface to the LICENSE REGISTER command.

Format: LICENSE REGISTER product-name

Example:

```
$ LICENSE REGISTER FORTRAN  /ISSUER=DEC  /AUTHORIZATION=USA-10 -
_$ /PRODUCER=DEC /UNITS=400 /VERSION=4.6 -
_$ /AVAILABILITY=F /CHECKSUM=1-HIDN-INDA-COMP-DAHH
```

```
$  LICENSE REGISTER  DVNETRTG /ISSUER=DEC -
_$ /AUTHORIZATION=USA-15 -
_$ /PRODUCER=DEC /UNITS=1000 /VERSION=4.0 -
_$ /AVAILABILITY=E /CHECKSUM=1-COOD-AGON-EFIC-HING
```

START

Sets up the license unit requirement table (LURT) for your system, and activates all licenses that are registered and enabled in the LICENSE database. Because the VMS operating system issues a LICENSE START command during system startup, you should need this command only if startup fails.

To use this command, you need the privilege to change mode to kernel (CMKRNL), the privilege to create system logical names (SYSNAM), and the system privilege (SYSPRV).

Format: LICENSE START

Example:

```
$ LICENSE START
```

This command sets up the LURT for your system and activates all the licenses that are registered and enabled in the LICENSE database.

UNLOAD

Deactivates a license, making the product unavailable from the current node. The product license or licenses must be registered in the LICENSE database and must have been previously activated with an interactive or automatic LICENSE LOAD command. The LICENSE UNLOAD command has no effect on active processes.

To use this command you need the privilege to change mode to kernel (CMKRNL), the privilege to create system logical names (SYSNAM), and the system privilege (SYSPRV).

Format: LICENSE UNLOAD product-name

Example:

```
$ LICENSE UNLOAD/PRODUCER=DEC FORTRAN
```

Messages

To ensure that LMF messages are displayed through the operator's communication facility (OPCOM), you must define the logical name LMF$DISPLAY_OPCOM_MESSAGE as follows:

```
$ DEFINE/EXEC/SYSTEM LMF$DISPLAY_OPCOM_MESSAGE TRUE
```

Reporting on User Activity

INTRODUCTION

This chapter presents the basic concepts used in user management. Among the topics covered are the steps used in collecting process information with the ACCOUNTING utility.

OBJECTIVE

To describe the tasks and responsibilities involved in managing users on a VMS system, a system and network manager should be able to use the ACCOUNTING utility to collect and report process information.

RESOURCE

- *VMS Accounting Utility Manual*

TOPICS

- Collecting process information with the ACCOUNTING utility
  - Using the ACCOUNTING utility to produce reports

COLLECTING PROCESS INFORMATION WITH THE ACCOUNTING UTILITY

The VMS system accounting file, **SYS$MANAGER:ACCOUNTNG.DAT**:

- Is created at system initialization
- Records system activity
- Records system resources used
- Provides information that allows you to analyze the relationship between system activity and performance
- Monitors system activity for security reasons

Manipulate the system accounting file using the following commands:

- SET ACCOUNTING
  - To close current accounting file and create a new version
  - To enable and disable the recording of specific system events
- SHOW ACCOUNTING
  - To display a list of system events for which accounting is enabled
- ACCOUNTING

JOB_CONTROL writes records to ACCOUNTNG.DAT when the following events occur:

- Process deletion/logout
- Print job completion
- Login failure
- Batch job completion

The contents of the accounting record include:

- System resource usage
- Identity of resource user

Use of the ACCOUNTING command requires read access to the input accounting file.

The full format accounting report (Example 7–1) is generated by:

```
$ ACCOUNTING /FULL /TYPE=PROCESS /PROCESS=INTERACTIVE
```

Example 7–1 Accounting Record, Full Format

```
INTERACTIVE Process Termination
-------------------------------

Username:          VAL              UIC:                [PERSONNEL,VAL]
Account:           PERSONNEL        Finish time:        30-APR-1991 17:46:43.23
Process ID:        212007CC         Start time:         30-APR-1991 10:15:36.40
Owner ID:                           Elapsed time:                 0 07:31:06.83
Terminal name:     VTA85:           Processor time:               0 00:11:37.22
Remote node addr:                   Priority:           4
Remote node name:                   Privilege <31-00>:  1014C000
Remote ID:                          Privilege <63-32>:  00000000
Queue entry:                        Final status code:  00000001
Queue name:
Job name:
Final status text: %SYSTEM-S-NORMAL, normal successful completion

Page faults:            65489       Direct IO:                4091
Page fault reads:        2383       Buffered IO:             28753
Peak working set:        1500       Volumes mounted:             0
Peak page file:          7226       Images executed:           379
```

Recording Accounting Information

Recording is enabled by default when the system is started. The **SET ACCOUNTING** command controls which types of records are written to ACCOUNTNG.DAT by JOB_CONTROL. By default, the accounting log file records each of the following activities for all users:

- BATCH - Batch job termination
- INTERACTIVE - Interactive job termination
- MESSAGE - User messages
- LOGIN_FAILURE - Login failures
- PRINT - Print jobs
- PROCESS - Process termination
- IMAGE - Image termination
- NETWORK - Network job termination
- SUBPROCESS - Subprocess termination
- DETACHED - Detached job termination

See Table 7–1 for information about using the SET ACCOUNTING command.

Table 7–1 Recording Accounting Information

Operation	Command Format and Examples
Enables the recording of all accounting information	`$ SET ACCOUNTING /ENABLE`
Disables the recording of all accounting information	`$ SET ACCOUNTING /DISABLE`
Enables the recording of accounting information selectively	`$ SET ACCOUNTING /ENABLE=(record-type[,...])` `$ SET ACCOUNTING /ENABLE=(PRINT,LOGIN_FAILURE)`
Disables the recording of accounting information selectively	`$ SET ACCOUNTING /DISABLE=(record-type[,...])` `$ SET ACCOUNTING /DISABLE=(LOGIN_FAILURE)`
Closes the current accounting file and opens a new one	`$ SET ACCOUNTING /NEW_FILE`

Image Accounting

Image accounting allows you to enable accounting for selected images when image accounting is disabled on the system.

Issue the following command for each image you would like to enable image-level accounting for:

```
$ INSTALL ADD file-spec /ACCOUNTING
```

When image accounting is enabled on the system, it logs entries for all images. The **/NOACCOUNTING** qualifier has no effect when image accounting is enabled.

Using the Accounting Utility to Produce Reports

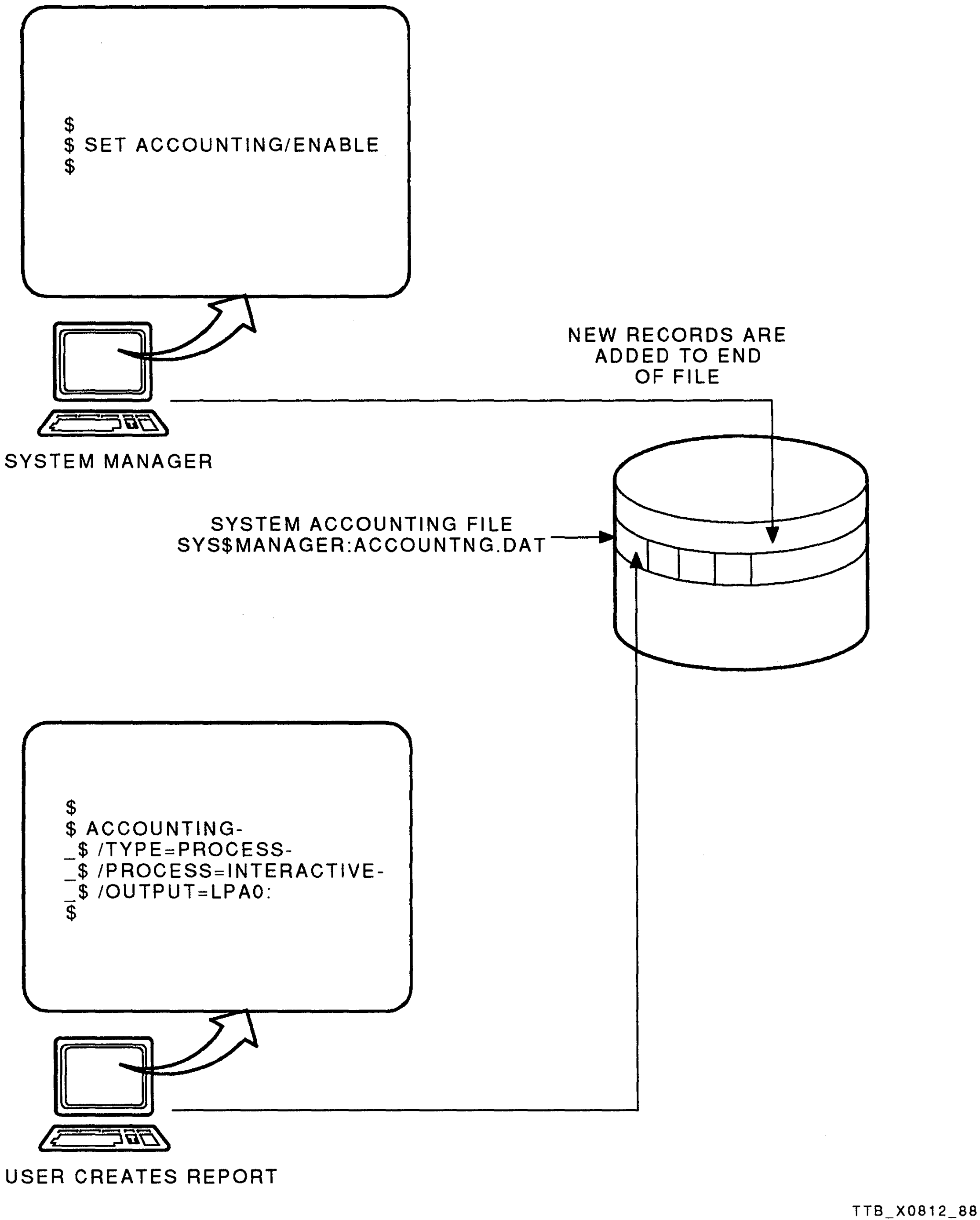
The ACCOUNTING utility reads the system accounting file and produces reports.

Qualifiers to the DCL ACCOUNTING command control:

- Accounting records to analyze
- Details to disclose:
  - Full
  - Brief
  - Summary
- Order in which to display records

See Figure 7–1 for an illustration of a system accounting file.

Figure 7–1 System Accounting File



The Brief Report Format

Suppose you have questions like:

> Did a particular user log in yesterday? If so, how many times? OR
> Which nodes did users set host from yesterday?

You can use a brief report like Example 7–2 to get the answer.

Example 7–2 Accounting Records, Brief Format

```
$ ACCOUNTING /SINCE=27-APR-1991:07:30 /BEFORE=27-APR-1991:08:30
     Date / Time      Type     Subtype      Username       ID     Source   Status
--------------------------------------------------------------------------------
27-APR-1991 07:32:00 PROCESS NETWORK     DECNET       20801B24 HARDY    10000004
27-APR-1991 07:42:47 PROCESS NETWORK     DECNET       20801AA5 SCDGAT   10000004
27-APR-1991 07:56:07 PRINT               BECKER       2080009B          00040001
27-APR-1991 08:01:57 PROCESS NETWORK     DECNET       20801A28 ZEKE     10000004
27-APR-1991 08:02:00 PROCESS NETWORK     DECNET       20801A29 HARDY    10000004
27-APR-1991 08:02:42 PROCESS NETWORK     DECNET       20801B2A HARDY    10000004
27-APR-1991 08:06:53 PROCESS INTERACTIVE KENT         2080182B VTA270:  10000001
27-APR-1991 08:09:37 PROCESS NETWORK     DECNET       20801AAC SCDGAT   10000004
27-APR-1991 08:12:23 PRINT               JOHNSTON     20201448          0000002C
27-APR-1991 08:13:48 PROCESS INTERACTIVE PIANTEDOSI   208017AF PARROT   10000001
27-APR-1991 08:15:03 PRINT               JOHNSTON     20201448          00040001
27-APR-1991 08:20:51 PROCESS NETWORK     DECNET       20801BAD SCDGAT   10000004
27-APR-1991 08:26:50 PROCESS NETWORK     DECNET       208015B0 UCOUNT   10000004
27-APR-1991 08:28:10 PRINT               BECKER       2080009B          00040001
```

The Summary Report Format

Suppose you wanted to answer a question like:

How much processor time did each interactive user consume? (This is something you may charge for or report to management on.)

Example 7–3 shows a Summary Accounting Report, highlighting processor usage.

Example 7–3 Summary Accounting Report for Processor Usage

```
$ ACCOUNTING /SINCE=05-JUN-1991:00:01 /BEFORE=06-JUN-1991 /TYPE=PROCESS -
_$ /SUMMARY=USER /REPORT=PROCESSOR

HH Username          Processor
                        Time
---------------------------------
02 OPERATOR          0 00:00:05.66
12 MOPPET            0 00:00:03.18
14 MOPPET            0 00:00:16.16
19 OPERATOR          0 00:08:28.60
```

How about each batch user? (You may charge a lower rate for batch users.)

Example 7–4 shows a Summary Accounting Report, highlighting processor usage for batch users.

Example 7–4 Summary Accounting Report, Processor Usage for Batch Users

```
$ ACCOUNTING /SINCE=05-JUN-1991 /BEFORE=06-JUN-1991 /TYPE=PROCESS -
_$ /PROCESS=BATCH /SUMMARY=USER /REPORT=PROCESSOR

Username          Processor
                     Time
------------------------------
DUFFY             0 00:52:54.19
JONES             0 00:00:02.99
MARSH             0 00:02:17.20
MORGAN            0 00:09:42.07
MOSTEIKA          0 00:00:03.95
OLIVEIRA          0 00:07:12.78
PARADISE          0 00:00:26.04
REGNELL           0 01:51:48.19
ROUNDS            0 01:01:22.35
SYSTEM            0 00:05:33.47
WTHOMAS           0 00:19:55.15
```

ACCOUNTING Qualifiers

Suppose you want to know how many print jobs a specific user printed in one day?

Example 7–5 is a summary Accounting Report that shows how many print jobs user MOPPET printed in one day.

Example 7–5 Accounting Report, Summary Print Information

```
$ ACCOUNTING /SINCE=05-JUN-1991:00:01 /BEFORE=06-JUN-1991:00:01 -
_$ /TYPE=PRINT /USER=MOPPET

     Date / Time       Type     Subtype     Username       ID      Source   Status
--------------------------------------------------------------------------------
 5-JUN-1991 09:28:00 PRINT                MOPPET       20203032            00000001
 5-JUN-1991 11:37:35 PRINT                MOPPET       20203032            00040001
 5-JUN-1991 13:41:22 PRINT                MOPPET       20203032            00000001
 5-JUN-1991 16:54:30 PRINT                MOPPET       20203032            00040001
```

Now suppose you wanted to know how many pages an average size print job for user MOPPET was?

Example 7–6 is a full Accounting Report entry that shows the page count of one of the print jobs user MOPPET printed that day.

Example 7–6 Accounting Report, Full Print Information

```
$ ACCOUNTING /SINCE=05-JUN-1991:00:01 /BEFORE=06-JUN-1991:00:01 -
_$ /TYPE=PRINT /USER=MOPPET /FULL

PRINT Job Termination
---------------------

Username:          MOPPET            UIC:                 [GROUP11,MOPPET]
Account:           VMS               Finish time:          5-JUN-1991 16:54:30.94
Process ID:        20203032          Start time:           5-JUN-1991 16:49:19.37
Owner ID:                            Elapsed time:                  0 00:05:11.57
Terminal name:                       Processor time:                0 00:00:00.00
Remote node addr:                    Priority:            100
Remote node name:                    Privilege <31-00>:   00000000
Remote ID:                           Privilege <63-32>:   00000000
Queue entry:       966               Final status code:   00040001
Queue name:        LPS40$SCDTST
Job name:          SYSNETII_SECURITY
Final status text: %JBC-S-NORMAL, normal successful completion

GETs from source:        4693
QIOs to printer:          288
Pages printed:             43
```

Suppose you wanted to know who has logged in to the operator's terminal during the last few days?

Example 7–7 shows that information.

Example 7–7 Accounting Report, Terminal Information

```
$ ACCOUNTING /SINCE=05-JUN-1991:00:01 /TERMINAL=OPA0:

     Date / Time      Type     Subtype     Username      ID      Source   Status
--------------------------------------------------------------------------------
 5-JUN-1991 02:15:41 LOGFAIL              <login>      2020241F OPA0:    10D38064
 5-JUN-1991 20:10:36 PROCESS INTERACTIVE OPERATOR     20204947 OPA0:    00030001
 5-JUN-1991 20:24:36 LOGFAIL              <login>      20204252 OPA0:    10D38064
 5-JUN-1991 21:55:57 LOGFAIL              <login>      2020495B OPA0:    10D38064
 6-JUN-1991 00:48:35 PROCESS INTERACTIVE OPERATOR     2020475C OPA0:    10010001
 6-JUN-1991 08:26:22 PROCESS INTERACTIVE STREET       20203F1D OPA0:    00000001
 6-JUN-1991 08:26:49 LOGFAIL              <login>      2020281F OPA0:    10D38064
```

Selecting Accounting Files

Example 7–8 shows how to select Accounting files.

Example 7–8 Selecting Accounting Files

```
$ ACCOUNTING /TYPE=PRINT
$
$ ACCOUNTING /TYPE=PRINT SYS$MANAGER:ACCO_91_JUN_12.DAT
$
$ ACCOUNTING /TYPE=PRINT SYS$MANAGER:ACCO_91_JUN*.DAT
```

A summary of some of the qualifiers that can be used to specify Accounting Report contents are highlighted in Table 7–2.

Table 7–2 Some Qualifiers Used to Specify Content of Accounting Report

Qualifier	Comments
/BEFORE=time	Selects records dated before specified time
/SINCE=time	Selects records dated after specified time
/QUEUE=queue-name	Name of print or batch queue
/JOB=job-name	Name of job sent to queue
/ENTRY=entry-number	Number generated when job was entered in queue
/PRIORITY=priority	Base priority of user – helps to create report on all interactive users or all real-time users
/ACCOUNT=account-name	Specified in UAF record
/UIC=uic	Specified in UAF record
/USER=user-name	Specified in UAF record
/TERMINAL=terminal-name	Device name of terminal
/PROCESS=process-type	BATCH, INTERACTIVE, DETACHED, and others
/TYPE=record-type	PRINT, LOGFAIL, PROCESS, and others

The qualifiers that you would use to affect the output format of the Accounting Reports are listed in Table 7–3.

Table 7–3 Qualifiers Affecting Output Format of Accounting Report

Qualifier	Comments
/TITLE=title	Specifies text to be printed at the top of report
/REPORT=item	Includes specified items in a summary report (default is REPORT=RECORDS)
/SORT=item	Sorts records in ascending or descending order by one or more items
/SUMMARY=item	To produce summary report, grouped by the items you specify in ascending order (default is USER)
/FULL	Displays all data in selected records. Do not use with /BINARY or /SUMMARY
/OUTPUT=file-spec	Sends the output to a specified file (default is SYS$OUTPUT)
/LOG	Displays log messages about progress of utility
/REJECTED=file-spec	Saves records NOT selected in a file in binary format
/BINARY	Produces output in binary rather than text format – useful for making a smaller accounting file from which to produce multiple reports

Creating an Accounting Report

There are six basic steps in creating an accounting report, shown in Table 7–4.

Table 7–4 Steps for Creating an Accounting Report

Step Operation	Command Element	Comment
1 Select an accounting file	Parameter	Default is SYS$MANAGER:ACCOUNTNG.DAT
2 Select the type of record to analyze	Qualifier /TYPE	Default is all types
3 Select records to analyze, based on the contents of specific fields in the records	Many qualifiers	Individual fields may be present in some record types and not in others
4 Sort the selected records	Qualifier /SORT	Affects display order of records in full and brief formats
5 Choose the format of the report	Qualifiers /FULL, /SUMMARY, /REPORT	Brief display is the default if no qualifiers are specified
6 Enter the appropriate command to produce the report	ACCOUNTING command	Requires read access to the accounting file

Using ACCOUNTING in Command Procedures

Use command procedures containing qualifiers to create accounting reports.

Full command string:

```
$ ACCOUNTING -
_$ /SINCE=02-APR-1991:05:00 -
_$ /BEFORE=02-APR-1991:10:00 -
_$ /TYPE=PROCESS -
_$ /SUMMARY=(HOUR,USER) -
_$ /REPORT=BUFFERED_IO
```

The contents of the DCL command procedure BUFFSUM.COM is:

```
/TYPE=PROCESS /SUMMARY=(HOUR,USER) /REPORT=BUFFERED_IO
```

Type the following to get the abbreviated command string incorporating the DCL command procedure BUFFSUM.COM

```
$ ACCOUNTING -
_$ /SINCE=02-APR-1991:05:00 -
_$ /BEFORE=02-APR-1991:10:00@BUFFSUM
```

SUMMARY

- The system accounting file is used to:
  - Record system activity
  - Charge for system resources used
  - Analyze relationship between system activity and performance
  - Monitor system activity for security reasons
- The information gathered in the system accounting file is then read and formatted by the ACCOUNTING utility. Input and output can be controlled by the system manager with DCL commands.

Maintaining System Security

INTRODUCTION

This chapter presents the basic concepts of maintaining system security. Among the topics covered are:

- The definition and identification of general system security
- An introduction to VAXcluster security considerations
- The conventions for using SYSGEN parameters and user quotas
- The steps used in a conversational startup

OBJECTIVES

To describe the tasks and responsibilities in maintaining system security, a system and network manager should be able to:

- Describe the VMS functions that are available for general system and VAXcluster security
- Assign privileges to users according to their needs
- Restrict user passwords by using SYSGEN parameters and other methods
- Perform a conversational startup

RESOURCES

- *VMS Authorize Utility Manual*
- *VMS DCL Dictionary*
- *Guide to VMS System Security*
- *VMS System Generation Utility Manual*
- *Guide to Maintaining a VMS System*
- *Guide to Setting Up a VMS System*
- *VMS VAXcluster Manual*
- *VMS Audit Analysis Utility Manual*
- *Hardware Operations Guide for VMS Systems*
- *VMS Installation and Operations Guide*

TOPICS

- Physical security
- Software security
  - VAXcluster security considerations
  - File Security
- Login security
  - System passwords
  - Filtering passwords
  - Password usage
  - Protecting terminals and other nonshareable devices
  - Password collection programs
  - Break-in detection at login
  - Intruder lists
  - Clearing intrusion records
- UIC and ACL protection
  - VMS protection using UICs
  - VMS protection using ACLs
- Tailoring user accounts
  - Access and security fields
  - Privileges
- Conversational startup

PHYSICAL SECURITY

Different installations require different levels of security. The system manager has control over the level of security implemented.

Security issues include:

- Physical security of computer
  - Access to console terminal
  - Availability of dial-up lines
- Media storage

SOFTWARE SECURITY

To ensure the basic minimum security, the system manager should:

- Keep SYSUAF.DAT, RIGHTSLIST.DAT, and NETPROXY.DAT up to date
- Use flags and hourly restrictions with AUTHORIZE
- Insist on nontrivial passwords
- Not publicize dial-up numbers
- Not permit WORLD access to SYSUAF.DAT
- Encourage use of file protection codes and ACLs
- Restrict user activity with Captive accounts
- Restrict user privileges
- Create accounting reports to check system usage
- Label disks and tapes in a systematic manner

For additional security, the system manager can include:

- Secondary passwords
- Erase-on-delete and erase-on-allocate for files
- Login security
- Break-in detection
- Security auditing
- Alarms on files

VAXcluster Security Considerations

A VAXcluster system should be treated as a single management domain. It must be managed as a single system, by a single system manager, or a cooperating management team. Even if a cluster has many different working environments, the different systems have access to common, shared resources and must be guided by a single security and management policy.

- Privileged users on one VAX node can affect the other nodes.
- There is no node-specific file protection by default.
  - The system manager can implement node-specific protection
  - Use a coordinated rights list and access control lists (ACLs)
- A network can provide greater security isolation between nodes than a VAXcluster environment.
- A single system implies that user names, UICs, and access rights are unique throughout the cluster.
- Multiple user authorization files
  - Not recommended in most cases
  - Break the single-system model of a cluster
  - Must be kept consistent manually
  - Do not isolate privileged users

File Security

Erase-On-Delete and Erase-On-Allocate

Erase-on-delete (EOD) refers to activity of the file system when files are deleted:

- File system overwrites blocks with zeros.
- Blocks are unavailable for allocation until overwritten.
- Users implement EOD on a file-by-file basis.
- Managers implement EOD on a volume-by-volume basis.

Erase-on-allocate (EOA), also known as high-water marking, refers to activity of the file system when files are created or extended:

- File system overwrites blocks with zeros (or pattern of your choice) before allocating them.
- Set automatically for each volume at initialization.
- For less security-sensitive volumes, disable high-water marking.

```
$ SET VOLUME/NOHIGHWATER_MARKING
```

NOTE

Enabling these system qualifiers does create an increase in system overhead.

Table 8–1 shows commands for setting erase-on-delete for a file or volume.

Table 8–1 Setting Erase-On-Delete for a File or Volume

Command/Qualifier and Example	Comments
`$ SET FILE/ERASE_ON_DELETE -` `_$ file-name`	Sets characteristic of file so file system performs an EOD when you delete it
`$ SET FILE/ERASE_ON_DELETE -` `_$ GOVERNMENT_SECRETS.DAT`	
`$ DELETE/ERASE_ON_DELETE -` `_$ file-name`	Tells the file system to perform an EOD as you delete the specified file or files
`$ DELETE/ERASE_ON_DELETE -` `_$ MY_SECRETS.DAT`	
`$ PURGE/ERASE_ON_DELETE -` `_$ file-name`	Tells the file system to perform an EOD on each file it deletes during the purge
`$ PURGE/ERASE_ON_DELETE -` `_$ COMPANY_SECRETS.*`	
`$ INITIALIZE/ERASE_ON_DELETE -` `_$ volume-name`	Tells the file system to perform an EOD for every file on this volume that users delete
`$ INITIALIZE/ERASE_ON_DELETE -` `_$ SECURE_VOLUME:`	
`$ SET VOLUME/ERASE_ON_DELETE -` `_$ SECURE_VOLUME:`	Modifies the volume's characteristic so the file system now performs an EOD on every file deleted from it

LOGIN SECURITY

Each user is normally assigned a user name and a password in the authorization file, SYSUAF.DAT. The user must enter both (unless the password is null) before using the system.

- Use the Authorize utility to set
  - Expiration date on UAF records
  - Minimum password length
  - Expiration date for password
  - Passwords that must be changed when user logs in for the first time
- Can define secondary password for accounts
- Can have the VMS system automatically provide a list of randomly generated passwords

You can define a secondary password for some accounts with the Authorize utility, as described in Table 8–2.

Table 8–2 Defining User Passwords

Commands	Comments
`$ RUN AUTHORIZE` `UAF> MODIFY SMITH /PASSWORD=MARY`	Modifies any user password with the Authorize utility.

`$ RUN AUTHORIZE`
`UAF> MODIFY SMITH`
`/PASSWORD=("",SECOND)`
	To add a secondary password to a UAF record, use the **/PASSWORD** qualifier with the **MODIFY** command. This command does not affect the current value of the primary password if you properly specify a null first password string.

`$ SET PASSWORD`	Modifies your own primary password.
`$ SET PASSWORD/SECONDARY`	Modifies your own secondary password.
`$ SET PASSWORD/GENERATE`	You can request or require (using the **/FLAG=GENPWD** qualifier in Authorize) that the VMS system generate a random list of passwords to choose from. If you enter the **/GENERATE** qualifier and receive a list, you must choose a password from that list, or request another list. (You do not need the **/GENERATE** qualifier if password generation has been set in your UAF record.)

System Passwords

For terminals in remote or sensitive locations, you can require a **system password** as shown in Table 8–3.

- Can require system password for terminals
- System password kept as a special UAF record
- System password can be set
  - System password can be initially set and changed

    Within AUTHORIZE: **UAF> MODIFY /SYSTEM_PASSWORD**
  - System password can be changed subsequently

    At DCL: **$ SET PASSWORD/SYSTEM**

Table 8–3 Defining a System Password for a Terminal

Step/Function	Comments

Step 1:
`$ SET PASSWORD/SYSTEM`
`Old password:`
`New password:`
`Verification:`
`$`
	Enter the old password in response to the first prompt. Then enter the new password in response to the next two prompts. None of the passwords are echoed while being entered. This command requires SECURITY and CMKRNL privileges.

Step 2:
`$ SET TERMINAL -`
`_$ /SYSPASSWORD /PERMANENT -`
`_$ TXA2`
	Set the specified terminal to require the system password. To log in to this terminal a user must press the RETURN key and enter the system password (no prompt is given). If successful, the user continues the normal login procedure when the Username: prompt appears. This command requires LOG_IO privilege.

Password Usage

Users and managers should be aware of password requirements for accounts and terminals.

Table 8–4 Using Passwords

Situation	Example	Comments

Normal account: requires one password	`RETURN`
	`Username: SMITH`
	`Password:`
		The system does not echo the password. Note that you can define the password to be null. If you do that, you do not receive any password prompt.

Account requiring primary and secondary passwords	`RETURN`
	`Username: SMITH`
	`Password:`
	`Password:`
		Passwords are not echoed. Typically, one person knows the primary password, but another knows the secondary one. Therefore, both must be present whenever this account is used.

Terminal requiring a system password	`Systempassword`
	`RETURN`
	`Username: SMITH`
	`Password:`
		You do not receive a prompt for the system password. User must hit the break key to begin the login process. To avoid excessive reports of broken terminals, make users aware of the ones requiring a system password, because unless they enter it successfully, the system will not display the Username: prompt.

Filtering Passwords

Filtering is a procedure for screening user-selected passwords for acceptability. Filtering is **not** on former system-generated passwords or passwords set in AUTHORIZE.

Dictionary Search

- New passwords are automatically checked against a *system password dictionary* to make sure the word is not present in the dictionary.
- The *system password dictionary* is kept in SYS$LIBRARY.
- /FLAGS=DISPWDDIC is the flag placed on user accounts (using the Authorize utility) to disable the dictionary search.

History Search

- The system keeps a list of 100 passwords used by each user and compares the new password with the list.
- SYS$SYSTEM:VMS$PASSWORD_HISTORY.DATA is where the password history list is stored.
- /FLAGS=DISPWDHIS is the flag placed on user accounts (using the Authorize utility) to disable the history search.

Protecting Terminals and Other Nonshareable Devices

Devices that are not shareable, such as terminals, have an owner UIC and a protection code that determine what processes can allocate the device.

- Terminals (nonshareable devices) have owner UIC and protection code that determine access
  - To control device ownership and protection

    ```
    $ SET PROTECTION/DEVICE
    ```

  - To place access control list (ACL) protection on terminals and tape drives

    ```
    $ SET DEVICE/ACL
    ```

Table 8–5 Establishing Ownership and Protection of Terminals and Other Nonshareable Devices

Operation	Format of SET PROTECTION/DEVICE Command and Examples (Requires OPER privilege)	Comments
Establishing protection ownership	`$ SET PROTECTION=code/DEVICE-` `_$/OWNER_UIC=[UIC] device`	By default, all terminals have the owner specified by the system parameter TTY_OWNER and the protection specified by the system parameter TTY_PROT.
Allowing all users access to a device	`$ SET PROTECTION/DEVICE device` `$ SET PROTECTION/DEVICE TTA3:`	The default value for TTY_PROT allows all users access to the device. If you do not specify a protection code with this command, you are assigning the default protection to the device.

Establishing system users as the owner	`$ SET PROTECTION =`
	`(S:R,O:R,G,W)-`
	`_$ /DEVICE/OWNER=[1,4] TTA3:`
		Only system users can allocate this terminal from a program. By specifying that Group and World users have no access, you protect the terminal against password collection programs run by users.

Password Collection Programs

You can protect terminals from password-collecting programs by setting the secure server characteristic on them:

```
$ SET TERMINAL/PERMANENT/SECURE_SERVER/DISCONNECT TTC1:
```

- Ensures that only the VMS login program receives user name and password.
- You must educate users to press the BREAK key (instead of RETURN key) to receive Username: prompt before logging in.
- The SECURE_SERVER characteristic has no effect on terminals with AUTOBAUD characteristic set.

Break-In Detection at Login

When you implement break-in detection, the VMS system records information about login failures in suspect lists.

- User name suspect list
- Terminal name suspect list
- Node name suspect list

Several SYSGEN parameters provide break-in control, as shown in Table 8–6.

Table 8–6 SYSGEN Parameters for Break-In Detection

Parameter	Comments
LGI_BRK_DISUSER	Once an intruder has been detected, the VMS system sets the DISUSER flag in the account's UAF record (if the parameter is set to 1). Manual intervention by the system manager is necessary to reactivate the account. Use this feature with caution. **Default = 0.**
LGI_BRK_LIM	Break-in limit defining the total number of consecutive login failures allowed within a reasonable time limit before a SUSPECT becomes an INTRUDER. **Default = 5.**
LGI_BRK_TERM	Controls the association of terminals and user names for counting failures. By default, the VMS system sets this parameter to 1 so that terminals and user names are tracked together. If you use terminal servers, then you might want to set this parameter to 0 (only track user names), since a LAT port (on the VAX side) is generally not a useful indication of the actual terminal being used. **Default = 1.**
LGI_BRK_TMO	Timeout factor expressed in seconds (a delta time). Used in conjunction with LGI_BRK_LIM to decide if a SUSPECT is an INTRUDER. The larger this value, the more secure your system. **Default = 300.**
LGI_HID_TIM	Time factor expressed in seconds. System uses this value in an equation to determine the time interval during which an INTRUDER is subject to evasive action. The time interval calculated using LGI_HID_TIM is different for each instance, so you never know exactly how long an INTRUDER will be evaded. **Default = (300*n)** where n = some number between 0 and 1.5.
LGI_RETRY_LIM	Limits the number of times a user can retry the login procedure when coming in through dial-up lines. **Default = 3.**
LGI_RETRY_TMO	The number of seconds allowed between login attempts on dial-up lines. **Default = 20.** This means the user must properly log in within 20 seconds of a failed attempt, or the system will hang up the line.

Suspects can gain access to the VMS system by entering the correct user name and password. Example 8–1 demonstrates a successful break-in.

Example 8–1 Break-In Suspect Logs in Successfully

```
❶ RETURN

❷           This is node TIDE

  Username: JONES

❸ Password:

  User authorization failure

❹ Username: JONES

  Password:

  User authorization failure

❺ Username: JONES

  Password:

            Welcome to node TIDE running V5.4

❻ Last interactive login on Saturday, 11-JUN-1990  08:59

❼        2 failures since last successful login

  $
```

Notes on Example 8–1:

❶ User presses the RETURN key.

❷ System displays the announcement message and Username: prompt.

❸ User enters an incorrect password and fails to log in.

❹ User presses RETURN again. This time the system displays no announcement message before displaying the Username: prompt. Two SYSGEN parameters govern whether the announcement message appears or not: LGI_RETRY_TMO and LGI_RETRY_LIM. The value of the first parameter determines how long the system waits between retries (default is 20 seconds). The value of the second parameter determines how many retries are allowed. These parameters affect dial-up users primarily because the system breaks their connection after these limits are reached, forcing the user to dial in again. The on-line user simply presses RETURN and receives the announcement message as well as the Username: prompt.

❺ User presses RETURN, logs in successfully and receives the welcome message and two other messages.

❻ The first message tells the user when someone last logged in successfully using that user name and password. If this message shows a login more recent than the last time this user actually logged in, the user should report it as someone may have guessed the user's password.

❼ The second message tells the user how many login attempts were made before the login was successful. If there is a difference between the number of failures reported and the number of times the user actually caused a failure, the user should report it as someone may be trying to break in to the system.

Intruder Lists

The VMS system creates intruder lists when login failures exceed the break-in limit, set by the system manager.

- Separate intruder lists for user name, terminal name, and node name
- Intruders subject to evasive action

Example 8–2 is an example of a break-in suspect becoming an intruder.

Example 8–2 Break-In Suspect Becomes an Intruder

```
  RETURN
               THIS IS NODE TIDE
  Username:  WEBSTER
1 Password:
  User authorization failure
  Username:  WEBSTER
  Password:
  User authorization failure
  Username:  WEBSTER
  Password:
  User authorization failure
  RETURN
2              THIS IS NODE TIDE
  Username:  WEBSTER
  Password:
3 User authorization failure
  Username:  WEBSTER
  Password:
4 User authorization failure
  Username:  WEBSTER
  Password:
  User authorization failure
```

Notes on Example 8–2:

1. The user enters an incorrect password on three login attempts and receives a user authorization failure message each time.

2. The user presses the RETURN key and receives the announcement message again because the LGI_RETRY_LIM parameter is set to 3 on this system. A dial-up user loses the carrier after the third unsuccessful attempt.

3. The user enters an incorrect password and receives an error message. If the password was correct, the user could still log in at this fourth attempt because the value of the LGI_BRK_LIM parameter on this system is 5.

4. The user enters the correct password, but since this is the sixth attempt, the system begins evasive action. (Evasive action includes responding to correct input with error messages for a variable period of time.) This user name cannot be used for a while to log in. If a user reports being unable to enter their account even after entering the correct password, an intruder might be trying to break in to the system. Change the password immediately and read the security audit reports on the console terminal.

Clearing Intrusion Records

The following example shows how to check for intrusion records, then how to delete the record from the intrusion database. Requires CMKRNL (Change Mode to Kernel) and SECURITY privileges.

```
$ SHOW INTRUSION
Intrusion         Type        Count  Expiration     Source
   TERMINAL       INTRUDER       6   04:49:02.02    _OPA0:
   TERM_USER      SUSPECT        5   20:39:27.38    _OPA0:MOPPET
$
$ DELETE/INTRUSION_RECORD _OPA0:MOPPET
$
$ DELETE/INTRUSION_RECORD _OPA0:
$
$ SHOW INTRUSION
%SHOW-F-NOINTRUDERS, no intrusion records match specification
```

UIC AND ACL PROTECTION

Table 8–7 ACL- and UIC-Based Protection

File Name and Owner UIC	Type of Protection	Access Allowed
FILE.DAT [200,011]	UIC-based protection: (S:RWED,O:RWED,G:RWE,W) No ACL set	System and Owner have RWED access. Group users have RWE access. World users have no access.
PROGRAM.FOR [200,011]	UIC-based protection: (S:RWED,O:RWED,G:RWE,W) (IDENTIFIER=SMITH,ACCESS=READ) ACL allows users holding the identifier name SMITH to read the file.	System and Owner have RWED access. Group users have RWE access. Users holding the identifier name SMITH have READ access only (even if they are in the Group category). Other users have no access.
JUNK.DAT [200,011]	UIC-based protection: (S:RWED,O:RWED,G:RWE,W) (IDENTIFIER=SMITH,ACCESS=NONE) ACL does not allow users holding the identifier name SMITH to have any access to the file.	System and Owner users have RWED access (even those holding the identifier name SMITH). Group users have RWE access except those holding the identifier name SMITH. World users have no access.

VMS Protection Using UICs

In assigning UICs to users, determine the extent to which user processes need to:

- Share access to files, volumes, and devices
- Communicate interactively with one other
- Affect or control one other

For greater interaction, place users in the same UIC group.

For greater protection, place users in different groups.

Table 8–8 suggests how to define groups based on user needs when you set up their UICs.

Table 8–8 Dividing Users into Groups

User Situation	Course of Action	Result
Users must allow some users access to their files and structures, but deny access to others on the system.	Divide users into groups along project or departmental lines. Assign a different group UIC number to each group. Leave room between each assigned number for additions later.	Users within a group can access each other's files with the access set for GROUP in the protection code of the accessed file. Users outside a group will be able to get access to files belonging to a group member with the access defined for WORLD only.
Users do not need to protect files against access by each other, but you must protect system files from their access. (System files usually have a group number from 0-10.)	Assign all users a UIC of [300,300] or some other UIC.[1]	All users own all user files and have OWNER access to all user files. Users cannot access system files.
Users need not protect files, but should divide them into groups for accounting and observation purposes.	Divide users into groups.[1] Either: 1. Modify the system parameter RMSFILEPROT to set the default protection assigned to all new files so WORLD has complete access. 2. Create a LOGIN.COM file for each user and include the **SET PROTECTION /DEFAULT=(W:RWED)** command in it. Do not include this command in your LOGIN.COM file if you are the system manager.	1. You can collect accounting information on separate users (according to their UICs) but since all files created allow all users complete access, the files are not really protected. In this case, newly created system files are not protected either. 2. You can collect accounting information on separate users (according to their UICs). Users can access all files on the system except your files and system files.

[1] Do not assign a group number reserved for system users (typically 0-10). Modify the MAXSYSGROUP parameter to set the upper limit of the group number for system users as needed.

Process Interaction

You can increase the ability of processes to interact and share information by assigning them UICs in the same group.

You can decrease this ability by assigning them UICs in different groups.

Table 8–9 details the effects of UIC group assignment on process interaction.

Table 8–9 Interaction Between Processes in Same Group

Interaction Available	Comment
Can share files, volumes, and devices, yet deny access to processes in other groups	Accomplished by setting the protection codes of the files, volumes, and devices properly
Can communicate with each other by means of structures that processes in other groups cannot access	For example: • Group logical names • Group mailbox logical names • Group global section names • Common event flag clusters
Can affect and control other processes in group	Requires GROUP privilege

VMS Protection Using ACLs

While VMS system protection using UICs is usually sufficient for most files on your system, the VMS operating system provides an additional layer of protection.

- Can be used to grant access to files for specific users rather than groups of users
- Based on **identifiers**
  - Users can hold one or more identifiers
  - Files can specify access rights for holders of various identifiers
- Record access information for files in an access control list (ACL)
- Define identifiers and who holds them in **rights database**
- When a user logs in, the VMS system creates an *access rights list* for that user, consisting of identifiers held
- When the user attempts to access files, the VMS system compares the access rights list of the user with the access control list of the file
  - If no ACL is on file, the VMS system determines access rights according to the UIC of the user and VMS protection code on file
  - If the ACL does not allow access, SYSTEM and OWNER users can still gain the type of access allowed them in the VMS protection code
  - If the user does not hold any identifiers listed in the file's ACL, the VMS system determines access rights according to the UIC of the user and VMS protection code on file

TAILORING USER ACCOUNTS

UAF record fields fall into four basic categories:

❶ Identification and environment

❷ Access and security

❸ Quotas and resource limits

❹ Privileges

Example 8–3 shows a sample listing of a UAF record and indicates the location of these field categories.

Example 8–3 UAF Record Field Categories

```
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: open file  20  Bytlm:       4096
  Maxacct jobs:     0  Shrfillm: shared file 0  Pbytlm: Meaningless  0
  Maxdetach:        0  BIOlm:          6  JTquota:     1024
  Prclm:            2  DIOlm:          6  WSdef:       1024
  Prio:             4  ASTlm:         10  WSquo:       2048
  Queprio: Meaningless 4  TQElm:      10  WSextent:    4096
  CPU:         (none)  Enqlm:         10  Pgflquo:    10000
❹ Authorized Privileges:
    GROUP TMPMBX NETMBX
  Default Privileges:
    TMPMBX NETMBX
```

Access and Security Fields

Access and security fields are used to:

- Limit the use of the account according to time/day or access mode
- Limit certain capabilities once logged in to the system
- Authenticate user requests for access to files and other resources

Access Times and Modes

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

- Time of day
- Day of week
- Access mode

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)
```

For example, the following commands allow SMITH to gain access to the system in any manner from 8 a.m. through 5 p.m. 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
```

Table 8–11 summarizes the AUTHORIZE qualifiers used to restrict access.

You can combine these values to further specify the account's ability to gain access to the system. For example, the following commands allow SMITH to use a local terminal during the day and a dial-up terminal during evenings and weekends.

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

Table 8–10 lists the various types of access modes.

Table 8–10 Login Access Modes

Mode	Description
INTERACTIVE	Any kind of interactive login
LOCAL	Directly connected terminals
DIALUP	Modem connections using telephone services (or network services emulating telephone services)
REMOTE	Virtual terminal connection across DECnet system
BATCH	Batch jobs (noninteractive access method)
NETWORK	DECnet noninteractive network access, for example: file transfers, electronic mail to/from other nodes, and so forth.

Table 8–11 summarizes the AUTHORIZE qualifiers used to restrict access.

Table 8–11 AUTHORIZE Qualifiers for Access Fields

Qualifier	Function
`/ACCESS[=(range[,...])`	Specifies hours of access for all modes of access
`/BATCH[=(range[,...])`	Specifies hours of access permitted for batch jobs
`/DIALUP[=(range[,...])` [1]	Specifies hours of access permitted for dialup jobs
`/INTERACTIVE[=(range[,...])`	Specifies hours of access permitted for interactive logins
`/LOCAL[=(range[,...])` [1]	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. Specifies primary days as MON, TUE, WED, THU, FRI, SAT, and SUN. Specifies secondary days as NOMON, NOTUE, NOWED, and so forth.
`/NETWORK[=(range[,...])`	Specifies hours of access permitted for network batch jobs
`/REMOTE[=(range[,...])` [1]	Specifies hours of access permitted for interactive logins initiated by network remote terminals

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

Login Flags

Login flags are used to restrict certain activities of the user's job. Table 8–12 is not a complete list; it lists login flag parameters that are related to security.

Table 8–12 Login Flag Parameters Related to Security

/FLAG Parameter [1]	Purpose
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
DISPWDDIC	Disables automatic screening of new passwords against a system dictionary
DISPWDHIS	Disables automatic checking of new passwords against a list of the user's passwords from the last year
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
RESTRICTED	Ensures that all commands are executed in the system and user login command procedures and any procedures invoked from these two procedures

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

NOTE

If your default account has the DISUSER flag set, you must add the /FLAG=NODISUSER qualifier when generating a new account.

Security Fields

Security fields are used to authenticate user requests for access to files and other resources. Table 8–13 lists the qualifiers used to set these fields, as well as control the rights database.

Table 8–13 AUTHORIZE Qualifiers for Security Fields

Qualifier	Function
`/ADD_IDENTIFIER`	Adds identifiers for the user name and account name to the rights database
`/EXPIRATION=time`	Expiration date and time of the account
`/GENERATE_PASSWORD[=keyword]`	Invokes the password generator to generate user passwords. Details of the possible keywords are discussed in the *VMS Authorize Utility Manual.*
`/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.

Table 6–3 lists the qualifiers used to set these and other process-related parameters.

Example 8–4 shows how you would change a user name and its corresponding identifier while leaving the UIC the same.

Example 8–4 Modifying User Name and Identifier

```
$ mc authorize
UAF> SHOW /ID MORGAN
  Name                              Value          Attributes
  MORGAN                            [000300,000020]
UAF> RENAME MORGAN BMORGAN/MODIFY_ID/PASSWORD=BMORGAN
%UAF-I-RENMSG, user record renamed
%UAF-I-RDBMDFYMSG, identifier MORGAN modified
UAF> SHOW /ID BMORGAN
  Name                              Value          Attributes
  BMORGAN                           [000300,000020]
UAF> EXIT
%UAF-I-DONEMSG, system authorization file modified
%UAF-I-NAFNOMODS, no modifications made to network proxy database
%UAF-I-RDBDONEMSG, rights database modified
```

Privileges

Two sets of privileges are specified in each UAF record:

- **Authorized**
  - Privileges enabled only by explicit use of the DCL command

    SET PROCESS/PRIVILEGE
- **Default**
  - Privileges automatically enabled once the user has logged in

Table 8–14 lists the qualifiers used to set an account's privileges.

Table 8–14 AUTHORIZE Qualifiers for Privilege Fields

Qualifier [1]	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.

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

A complete listing of VMS privileges is presented in Table 8–15.

Detailed definitions of these privileges may be found in the *Guide to Setting Up a VMS System*.

Privileges are divided into seven categories according to the damage that the user possessing them could cause the system:

- None - No privileges
- Normal - Minimum privileges to effectively use the system
- Group - Potential to interfere with members of the same group
- Devour - Potential to consume noncritical system-wide resources
- System - Potential to interfere with system operation
- File - Potential to compromise file security
- All - Potential to control the system

Table 8–15 VMS Privileges

Category	Privilege	Allows User to:
None	None	None requiring privileges
Normal	MOUNT	Execute mount ACP function
	NETMBX	Create network device
	TMPMBX	Create temporary mailbox
Group	GROUP	Affect other processes in same group
	GRPPRV	Have group access by system protection
Devour	ACNT	Suppress accounting message
	ALLSPOOL	Allocate spooled device
	BUGCHK	Make bugcheck log entries
	EXQUOTA	Exceed quota
	GRPNAM	Insert in group logical name table
	PRMCEB	Create permanent common event clusters
	PRMGBL	Create permanent global sections
	PRMMBX	Create permanent mailbox
	SHMEM	Create/delete objects in shared memory

Table 8–15 VMS Privileges (Cont)

Category	Privilege	Allows User to:
System	ALTPRI	Set any priority value
	OPER	Have operator privilege
	PSWAPM	Change process swap mode
	SECURITY	Perform security functions
	SYSLCK	Lock system-wide resources
	WORLD	Affect other processes in the world
Files	DIAGNOSE	Diagnose devices
	SYSGBL	Create system-wide global sections
	VOLPRO	Override volume protection
All	BYPASS	Bypass UIC checking
	CMEXEC	Change mode to executive
	CMKRNL	Change mode to kernel
	DETACH	Create detached processes
	LOG_IO	Do logical I/O
	PFNMAP	Map to specific physical pages
	PHY_IO	Do physical I/O
	READALL	Read anything as the owner
	SETPRV	Set any privilege bit
	SHARE	Assign channels to nonshared device
	SYSNAM	Insert in system logical name table
	SYSPRV	Access objects by system protection

CONVERSATIONAL STARTUP

The VMS system includes the option of examining and modifying the system parameters during startup, before the processor uses them to customize the system. This option is called a **conversational startup**.

During a conversational boot a system manager can:

- Specify a minimum startup
- Select alternate file as the source of system parameter values
- Set and show individual parameter values
- Specify an alternate site-independent startup procedure
- Turn on verification during startup command procedure execution

For a system-specific detailed description, refer to the *Hardware Operation Guide for VMS Systems*.

Table 8–16 Using SYSBOOT During Conversational Startup

Function	Command Format and Examples

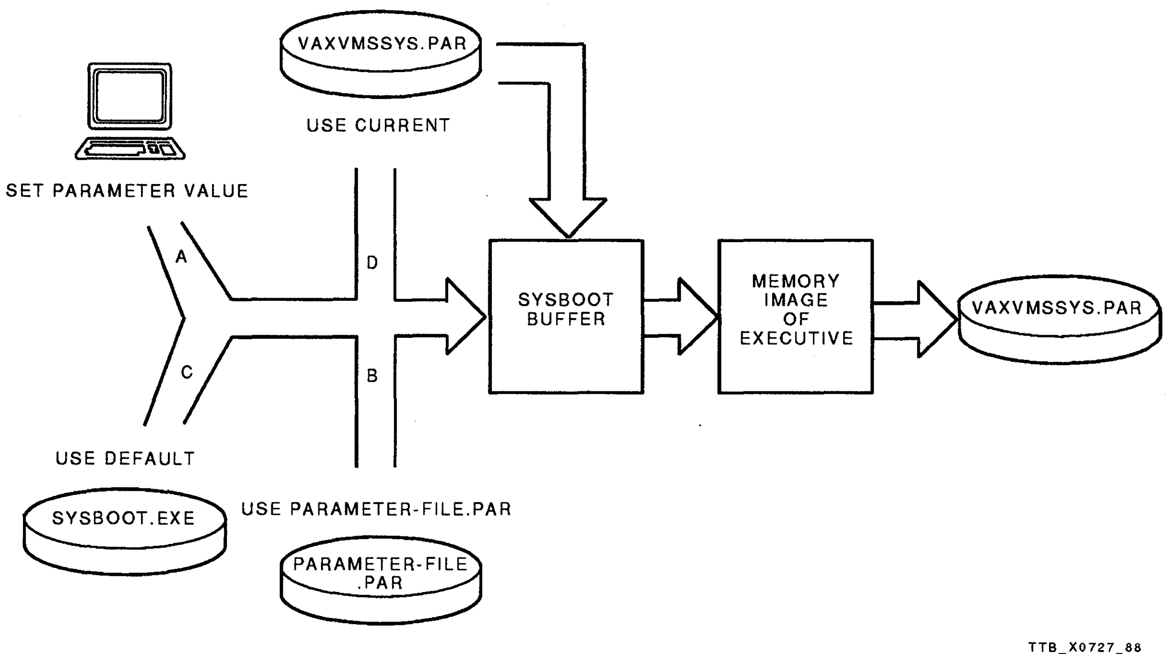
Examines a system parameter or group of parameters	`SYSBOOT> SHOW parameter`
	`SYSBOOT> SHOW /parameter-group`
	`SYSBOOT> SHOW MAXPROCESSCNT`
	`SYSBOOT> SHOW /ALL`

Modifies a system parameter	`SYSBOOT> SET parameter value` `SYSBOOT> SET MAXPROCESSCNT 60`

Modifies a group of system parameters (.PAR files should be in SYS$SYSTEM)	`SYSBOOT> USE parameter-file.PAR`
	`SYSBOOT> USE CURRENT`
	`SYSBOOT> USE DEFAULT`
	`SYSBOOT> USE ALTPARAM.PAR`

Uses an alternate DCL startup file	`SYSBOOT> SET/STARTUP SYS$SYSTEM:startup-file` `SYSBOOT> SET/STARTUP SYS$SYSTEM:ALTSTART`
Exits SYSBOOT to continue the startup procedure	`SYSBOOT> CONTINUE`

Figure 8–1 Effect of VMS Startup on System Parameters



Bypassing the User Authorization File

When you have a locked system, such as when all passwords have been forgotten, one method of breaking into the system is to set the alternate user authorization file.

This method requires setting the system parameter UAFALTERNATE, which defines the logical name SYSUAF to refer to the file SYS$SYSTEM:SYSUAFALT.DAT.

Use the following procedure to set the alternate user authorization file:

1. Using the instructions that can be found in the *VMS Installation and Operations* supplement for your computer, perform a conversational boot. (Follow the instructions only to where you receive the SYSBOOT> prompt.)

2. At the SYSBOOT> prompt, enter the following command:

   ```
   SYSBOOT> SET UAFALTERNATE 1
   ```

   NOTE

   If your system is running DECwindows software, you must disable the windowing system by entering the following command:

   ```
   SYSBOOT> SET WINDOW_SYSTEM 0
   ```

3. Type CONTINUE and press Return.

4. After the startup procedure completes, you will be prompted for USERNAME: and PASSWORD:. You should then log in on the console terminal by entering any user name and password.

5. Fix the problem that caused you to be locked out of the system. If it was a forgotten password, you should make the necessary changes to the UAF.

   ```
   $ DEFINE/SYSTEM/EXECUTIVE_MODE SYSUAF SYS$SYSTEM:SYSUAF.DAT
   $ SET DEFAULT SYS$SYSTEM
   $ RUN AUTHORIZE
   At the UAF> prompt modify the necessary passwords.
   ```

6. Restore the UAFALTERNATE parameter using SYSGEN. If necessary, restore the WINDOW_SYSTEM parameter back to its original value.

   ```
   $ RUN SYS$SYSTEM:SYSGEN
   SYSGEN> SET UAFALTERNATE 0
   SYSGEN> SET WINDOW_SYSTEM 1   (If necessary)
   SYSGEN> WRITE CURRENT
   SYSGEN> EXIT
   ```

7. The final step is to shut down the system and then reboot.

SUMMARY

- The system must be made secure in two ways, both physically and its software.
- One of the most important ways of securing the software is through strict login security:
  - Using system passwords
  - Screening a user-selected password for acceptability
  - Protecting terminals and other nonshareable devices
  - Monitoring against password collection programs and other break-in attempts
- System managers must set up user accounts to provide the security needed for the environment they are working in. The VMS operating system provides a UIC-based protection but also an additional layer of ACL protection if you choose to enable it.
- System managers can tailor user accounts to specify when and where a user can log in. They may also grant users a full range of privileges to allow precise access and control of various resources and operating system facilities.
- VMS provides the SYSBOOT conversational startup utility as one method to modify system parameters.

Managing a Network Node

INTRODUCTION

Once the DECnet software has been configured, the network runs automatically and requires no intervention.

The few cases in which the system manager needs to perform day-to-day management of a DECnet node include:

- Adding node entries to the remote node database as new nodes are added to the network
- Removing entries as nodes are removed from the network

This chapter introduces the DECnet permanent and volatile databases and the utility that manages them, the network control program (NCP). It presents specific NCP operations that are used to:

- Create, display, and modify node database parameters
- Obtain node information from a remote node
- Obtain a complete node database from another node in the network

OBJECTIVES

To monitor and control the network, a system and network manager should be able to:

- Maintain the local node database
- Obtain node information from other nodes

RESOURCES

- *Guide to DECnet–VAX Networking*
- *VMS Networking Manual*
- *VMS Network Control Program Manual*

TOPICS

- Review: starting, stopping, and monitoring the network
- NCP overview
- Maintaining the remote node database
- Executing remote commands

REVIEW: STARTING, STOPPING, AND MONITORING THE NETWORK

Starting the Network

This command starts DECnet software and should be included in the SYSTARTUP_V5.COM procedure:

```
$ @SYS$MANAGER:STARTNET.COM
```

Stopping the Network

The following commands shut down the network, allowing existing network operations to complete but preventing new operations from being started:

```
$ RUN SYS$SYSTEM:NCP
NCP> SET EXECUTOR STATE OFF
NCP> [CTRL/Z]
```

Or SHUT better allows existing and commands to complete

The SHOW NETWORK Command

The SHOW NETWORK command tells you whether the DECnet software is running.

On a nonrouting node:

```
$ SHOW NETWORK
VAX/VMS Network status for local node 62.820 TOYDOC on 11-SEP-1991 16:03:09.56

This is a nonrouting node, and does not have any network information.
The designated router for TOYDOC is node 62.1022 ZKDRC.
```

On a routing node:

```
$ SHOW NETWORK

VAX/VMS Network Status for local node 2.161 ARAKIS on
    19-APR-1990 09:18:03.07
The next hop to the nearest area router is node 2.62 ZEUS.
    Node        Links  Cost   Hops  Next Hop to Node
2.161  ARAKIS     0     0      0     Local    ->   2.161  ARAKIS
2.1    RAEL       0     8      1     UNA-0    ->   2.1    RAEL
    .
    .
    .
2.63   AURORA     0     8      1     UNA-0    ->   2.63   AURORA
                      Total of 7 nodes.
```

If the DECnet software is not running:

```
$ SHOW NETWORK
%SHOW-I-NONET, network unavailable
```

DECnet CONFIGURATION DATABASES

A DECnet configuration database contains data about the network, including:

- *Nodes* — other systems connected to the network
- *Objects* — network applications on the local node
- *Circuits* — communication paths between nodes
- *Lines* — physical data paths between nodes

The DECnet configuration database consists of two distinct databases, one permanent and one volatile.

Permanent Database

The permanent database is a collection of files in SYS$SYSTEM that contain the permanent settings for the network parameters. Features of the permanent database:

- Resides on disk
- Provides initial values for the volatile database
- Changes take effect the next time DECnet software is started up

Volatile Database

The volatile database is a memory-resident image containing current information about network management components. Features of the volatile database are:

- Allows changes to be made to a running system
- Changes take effect immediately
- Values are lost when system is shut down

Maintaining the DECnet Configuration Database

- Most of the information in the DECnet configuration database is set automatically when the node is installed in the network.
- You may need to maintain some of the information manually:
  - Change the name and address of the local node.

    The executor (local) database is SYS$SYSTEM:NETNODE_LOCAL.DAT.
  - Update the list of names and addresses of the remote nodes in the network.

    The remote node database is SYS$SYSTEM:NETNODE_REMOTE.DAT.

  You do not edit the databases directly; instead you use the network control program (NCP).

Identifying a Node

Each node in the network is identified by two values:

- The node *name*, which consists of up to six alphanumeric characters
- The node *address*, in the form **aa.nnnn**, where:
  - **aa** is the area number (from 1 to 63, or 0 if the network is not divided into areas)
  - **nnnn** is the node number within the area (from 1 to 1024)
- Every node in the network must have a unique name and address.
  - Someone must act as a *network administrator*, assigning names and addresses to guarantee that they are unique.

NCP OVERVIEW

As nodes are added to and removed from the network, use the network control program (NCP) to maintain the remote node database on your system.

Invoking and Exiting NCP

To invoke NCP:

```
$ RUN SYS$SYSTEM:NCP
NCP>
```

Table 9–1 shows the commands that you use to maintain the configuration databases.

Table 9–1 Commands that Maintain the Configuration Databases

Function	Command for Volatile Database	Command for Permanent Database
Create/modify parameters	SET	DEFINE
Delete parameters	CLEAR	PURGE
Display parameters	SHOW	LIST

DECnet Privileges

You must be logged in to a privileged account to access the permanent database or modify the volatile database. Table 9–2 lists the privileges required for various NCP commands.

Table 9–2 Required DECnet Privileges

Command(s)	Privilege(s)	Reason
SHOW	None	SHOW does not affect either database, nor does it access the permanent database.
SET, CLEAR	OPER	These commands affect the volatile database only, but must still be used with caution.
LIST	SYSPRV	The LIST command displays information from the permanent database. Only users with system privilege may access the permanent database. To display password information, BYPASS is needed as well.
DEFINE, PURGE	OPER, SYSPRV	These commands manipulate the permanent database. OPER is needed to manipulate any database; SYSPRV is needed to access the permanent database.

MAINTAINING THE REMOTE NODE DATABASE

Adding Remote Nodes to Your Configuration Database

Example 9–1 shows how to add a remote node to the configuration database.

Example 9–1 Adding a Remote Node

```
NCP>SHOW KNOWN NODES ❶

Known Node Volatile Summary as of  9-SEP-1988 11:23:44

Executor node = 26.60 (LUIGI)

State                    = on
Identification           = DECnet-VAX V5.0,  VMS V5.0

    Node            State      Active  Delay   Circuit     Next node
                               Links

 4.20  (FIGMEN)                                QNA-0        26.15  (CYCLPS)
24.29  (ANCHOR)                                QNA-0        26.15  (CYCLPS)
26.15  (CYCLPS)                                QNA-0        26.15  (CYCLPS)
26.24  (DEMON)                                 QNA-0        26.15  (CYCLPS)
26.25  (SPRITE)                                QNA-0        26.15  (CYCLPS)
26.49  (PILGRM)                                QNA-0        26.15  (CYCLPS)
26.61  (ENT)                      1       1    QNA-0        26.15  (CYCLPS)
26.130  (SWSVAX)                               QNA-0        26.15  (CYCLPS)
26.143  (PARROT)                               QNA-0        26.15  (CYCLPS)
26.148  (TBD3)                                 QNA-0        26.15  (CYCLPS)

NCP>DEFINE NODE 3.4 NAME TOYS ❷

NCP>SET NODE 3.4 ALL ❸

NCP>SHOW KNOWN NODES

Known Node Volatile Summary as of  9-SEP-1988 11:24:09

Executor node = 26.60 (LUIGI)

State                    = on
Identification           = DECnet V5.0,  VMS V5.0

    Node            State      Active  Delay   Circuit     Next node
                               Links

 3.4   (TOYS)                                  QNA-0        26.15  (CYCLPS) ❹
 4.20  (FIGMEN)                                QNA-0        26.15  (CYCLPS)
24.29  (ANCHOR)                                QNA-0        26.15  (CYCLPS)
26.15  (CYCLPS)                                QNA-0        26.15  (CYCLPS)
26.24  (DEMON)                                 QNA-0        26.15  (CYCLPS)
26.25  (SPRITE)                                QNA-0        26.15  (CYCLPS)
26.49  (PILGRM)                                QNA-0        26.15  (CYCLPS)
26.61  (ENT)                      1       1    QNA-0        26.15  (CYCLPS)
26.130  (SWSVAX)                               QNA-0        26.15  (CYCLPS)
26.143  (PARROT)                               QNA-0        26.15  (CYCLPS)
26.148  (TBD3)                                 QNA-0        26.15  (CYCLPS)
```

Notes on Example 9–1:

❶ The SHOW KNOWN NODES command confirms that neither the name TOYS nor the address 3.4 is already in use (in the volatile database).

❷ The DEFINE NODE command adds the node to the permanent database.

The command DEFINE NODE TOYS ADDRESS 3.4 would be equivalent.

❸ The SET NODE ... ALL command copies the value in the permanent database into the volatile database.

❹ The second SHOW KNOWN NODES command confirms that the node was added with the correct name and address.

Note that DECnet software automatically determines which circuit is used for communication with the remote node, and which node is the next one in the path to that remote node.

Copying Known Nodes

As new nodes are added to the network, it is each network/system manager's responsibility to maintain the list of remote nodes in the database. The system manager on the local node can:

- Manually enter a separate command for each node that needs to be defined
- Use the COPY KNOWN NODES command

The COPY KNOWN NODES command copies the names and addresses of remote nodes from a remote database to your local node's database. Follow these steps to use the COPY KNOWN NODES command:

1. Add a node to your database that has a complete list of node definitions.

   ```
   NCP>SET NODE 2.4 NAME COOKIE
   ```

2. Then, to copy the remote node information from node COOKIE to both the permanent and volatile databases:

   ```
   NCP>COPY KNOWN NODES FROM COOKIE TO BOTH
   ```

3. Use the SHOW KNOWN NODES or LIST KNOWN NODES command to ensure that the remote nodes are defined on your system.

Removing a Node from the Database

Sometimes it is necessary to remove information about a remote node from your remote node database. Use the CLEAR and PURGE commands to accomplish this.

Example 9–2 demonstrates how to remove a node from the volatile database.

Example 9–2 Removing a Node from the Database

```
NCP>SHOW NODE ZODIAC ❶

Node Volatile Summary as of 26-SEP-1988 16:28:10

    Node           State      Active  Delay   Circuit     Next node
                              Links

26.190 (ZODIAC)                                QNA-0        26.133
NCP>
NCP>CLEAR NODE ZODIAC ALL ❷
NCP>
NCP>SHOW NODE ZODIAC ❸

Node Volatile Summary as of 26-SEP-1988 16:28:30

%NCP-W-UNRCMP, Unrecognized component , Node
```

Notes on Example 9–2:/bold

❶ First, make sure the node is in the database using a SHOW NODE command.

❷ The CLEAR NODE command removes the node entry from the volatile database. If the node is also in the permanent database, you must use a PURGE NODE command to remove it.

❸ A second SHOW NODE command confirms that the node is no longer in the volatile database.

Changing Remote Node Entries

You can change a remote node's node name or node address by first deleting the remote node from your node's database and then redefining the remote node.

If node WENDI moves to a different location on the network, you can change its address in your database by issuing the commands shown in Example 9–3.

Example 9–3 Changing a Remote Node Entry

```
NCP>SHOW NODE WENDI ❶

Node Volatile Summary as of 27-SEP-1988 10:18:00

    Node           State      Active  Delay   Circuit     Next node
                              Links

26.89 (WENDI)                                 QNA-0        26.133
NCP>
NCP>CLEAR NODE WENDI ALL ❷
NCP>SET NODE 4.118 NAME WENDI ❸
NCP>SHOW NODE WENDI

Node Volatile Summary as of 27-SEP-1988 10:19:02

    Node           State      Active  Delay   Circuit     Next node
                              Links

 4.118 (WENDI)                                QNA-0        26.133 ❹
NCP>
```

Notes on Example 9–3:

❶ First find out if the node address is current.

❷ If the node address is not current, remove the node from the database.

❸ Then, add the node with the correct address.

❹ Another SHOW command confirms that the node is now defined correctly.

EXECUTING REMOTE COMMANDS

Most NCP commands issued on the local node are executed on that node. Occasionally, you may want to issue commands from the local node to be executed on remote nodes.

- The executor node is the node on which NCP functions are actually performed.
- To perform network management functions on remote nodes, NCP supports two commands:
  - TELL
  - SET EXECUTOR NODE

TELL Command

Use the TELL command to:

- Execute a single command at a remote node
- Temporarily override the current executor

Example 9–4 shows the use of the TELL command.

Example 9–4 Using the TELL Command

```
NCP>CLEAR EXECUTOR NODE ❶
NCP>SHOW EXECUTOR ❷

Node Volatile Summary as of 29-SEP-1988 18:53:01

Executor node = 26.60 (LUIGI)

State                    = on
Identification           = DECnet-VAX V5.0,  VMS V5.0

NCP>TELL PARROT SHOW EXECUTOR ❸

Node Volatile Summary as of 29-SEP-1988 18:53:18

Executor node = 26.143 (PARROT)

State                    = on
Identification           = DECnet-VAX V5.4,  VMS V5.4
Active links             = 4

NCP>TELL PARROT SHOW NODE LDYBUG ❹

Node Volatile Summary as of 29-SEP-1988 19:44:40

    Node             State      Active  Delay   Circuit     Next node
                                Links

 5.571  (LDYBUG)                                DMP-0        26.221 (AP026B)
```

Notes on Example 9–4:

❶ The CLEAR EXECUTOR NODE command ensures that the executor node is the local node, LUIGI (in case you previously entered a SET EXECUTOR command).

❷ The SHOW EXECUTOR command shows which node is currently the executor.

❸ The TELL command tells NCP to use the remote node's network management software and databases to execute the SHOW EXECUTOR CHARACTERISTICS command. The output indicates that the node executing the command was PARROT.

❹ A good use of the TELL command is to find out the address or name of a node that is not in the database on the local system, but is in the database on another system.

Setting an Executor Node

An alternative to using multiple TELL commands is the SET EXECUTOR NODE command. It allows you to choose the node at which you want NCP commands to execute, as shown in Example 9–5.

Example 9–5 Setting an Executor Node

```
NCP>SHOW EXECUTOR ❶

Node Volatile Summary as of 9-SEP-1988 14:57:39

Executor node = 26.60 (LUIGI)

State                        = on
Identification               = DECnet-VAX V5.0, VMS V5.0

NCP>SET EXECUTOR NODE BOSTON ❷
NCP>SHOW EXECUTOR ❸

Node Volatile Summary as of 9-SEP-1988 14:57:55

Executor node = 2.7 (BOSTON)

State                        = on
Identification               = DECnet-VAX V5.0, VMS V5.0

NCP>CLEAR EXECUTOR NODE ❹
NCP>SHOW EXECUTOR

Node Volatile Summary as of 9-SEP-1988 14:58:34

Executor node = 26.60 (LUIGI)

State                        = on
Identification               = DECnet-VAX V5.0, VMS V5.0
```

Notes on Example 9–5:

❶ Before the executor node is manipulated, the local node (LUIGI) is the executor node.

❷ The SET EXECUTOR NODE command temporarily changes the executor node to BOSTON. All subsequent commands will be executed on BOSTON until the executor node is changed again.

❸ The SHOW EXECUTOR command displays information about the current executor node.

❹ The CLEAR EXECUTOR NODE command resets the executor node back to the local node (LUIGI). The subsequent SHOW EXECUTOR NODE confirms this.

Indicating Access Control for Remote Command Execution

With the SET EXECUTOR NODE command, you can specify access control information as shown in Example 9–6.

Use this feature if a TELL or SET EXECUTOR command results in a "login information invalid at remote node" message.

Example 9–6 Using Access Control for Remote Command Execution

```
NCP>SET EXECUTOR NODE LUIGI"SYSTEM CHOCOLATE"
NCP>LIST EXECUTOR CHAR

Node Permanent Characteristics as of 29-SEP-1988 18:34:42

Executor node = 26.60 (LUIGI)

Management version        = V4.0.0
Type                      = nonrouting IV
Maximum address           = 1023
Nonprivileged user id     = DECNET
Nonprivileged password    = DECNET
```

SUMMARY

Every DECnet node has a configuration database that defines the characteristics of that node and determines how it functions within the network. In VMS implementations, this configuration database is provided within the DECnet software supplied by Digital.

To provide network management flexibility, each node's database consists of two distinct databases:

- Permanent (fixed) database
- Volatile (temporary) database

The network control program (NCP) is a network management tool to create, display, and modify component parameters in the DECnet configuration databases. NCP is used to:

- Monitor and test the network
- Create, display, and modify permanent and volatile database parameters for DECnet software

Node, line, and circuit commands help network managers to configure a node properly in the DECnet network. The SET/DEFINE NODE command is used to add a remote node to your system's configuration database. The COPY KNOWN NODES command eliminates the need to manually add entries for remote nodes to the database. This command copies the remote node entries from another node's configuration database.

For added flexibility in network management, NCP incorporates the concept of an executor node. The executor node is the node on which NCP functions are actually performed. To perform network management functions on remote nodes, NCP supports two commands:

- TELL
- SET EXECUTOR NODE

System Monitoring

INTRODUCTION

Monitoring the system is one of the primary responsibilities of the system manager. It allows the system manager to ascertain the performance of the system, while providing a characteristic view of the normal activity. This is an asset when problems arise. The MONITOR utility and particular DCL SHOW commands are discussed to help system managers interpret or analyze the various status/performance reports and graphs. Discussion also includes how to monitor a network and VAXcluster system, since most VAX systems today form that popular configuration.

Memory is often the key to improving system performance, provided it is utilized correctly. Monitoring memory resource usage will be helpful in determining system performance, which will be introduced later in the *VMS System and Network Management III* course. While it is often hard to divorce the two subjects, monitoring the system, and the tools or commands used to monitor it, is the focus of this chapter, not system performance.

OBJECTIVES

System managers should periodically monitor system activity in order to establish a baseline of system performance. To do this effectively, they should be able to:

- Monitor process memory usage
- Monitor process paging demands on the system
- Utilize the MONITOR utility for obtaining this specific process information
- Utilize appropriate DCL commands to help with analysis of memory resource usage
- Monitor a network and utilize specific network control program (NCP) commands
- Monitor the VAXcluster system by using various SHOW CLUSTER and MONITOR utility commands

RESOURCES

- *Introduction to VMS System Management*
- *VMS System Manager's Manual*
- *VMS Monitor Utility Manual*
- *VMS Network Control Program Manual*
- *Guide to DECnet–VAX Networking*
- *VMS Networking Manual*
- *VMS Show Cluster Utility Manual*

TOPICS

- MONITOR Utility
- Monitoring with DCL commands
- Using the network control program (NCP)
- Using SHOW CLUSTER in a VAXcluster system

MONITOR UTILITY

To display information about system resource usage:

```
$ MONITOR class-name(s) [/qualifiers]
```

Example 10–1 Invoking the MONITOR Utility

```
$ MONITOR
MONITOR>
```

MONITOR commands can:

- Display a class of information
- Set default classes
- List defaults
- Execute command procedures
- Obtain help
- Exit from the utility

Unlike most utilities that display system information, MONITOR can:

- Display several classes of information alternately
- Summarize statistics over a long period of time
- Record information in a disk file
- Play back information that it has recorded

Example 10–2 displays the MONITOR SYSTEM screen.

Example 10–2 MONITOR SYSTEM Screen Display

```
Node: OTHER                 VAX/VMS Monitor Utility     27-AUG-1991 17:18:41
Statistic: CURRENT             SYSTEM STATISTICS

                                                    Process States
          + CPU Busy (96)            -+        LEF:      5       LEFO:      0
          |***********************   |        HIB:     19       HIBO:      0
CPU     0 +--------------------------+ 100    COM:      2       COMO:      0
          |**************            |        PFW:      0       Other:     1
          +--------------------------+        MWAIT:    0
          Cur Top: BATCH_1036 (56)                     Total: 27

          + Page Fault Rate (108)    -+        + Free List Size (76414)   -+
          ||*************************|        |**********************    | 89K
MEMORY  0 +--------------------------+ 100  0 +--------------------------+
          |**************************|        |*********                 | 2621
          +--------------------------+        + Modified List Size (917) +
          Cur Top: MARCH (106)

          + Direct I/O Rate (52)     -+        + Buffered I/O Rate (4)    -+
          |**********************    |        |                          |
I/O     0 +--------------------------+ 60   0 +--------------------------+ 150
          |********************      |        |                          |
          +--------------------------+        +--------------------------+
          Cur Top: BATCH_1036 (47)            Cur Top: MARCH (4)
```

- The display is updated regularly (every three seconds by default)
- Each value is averaged over the sampling interval (three seconds by default) that yields a rate on a per second basis
- For certain displays (SYSTEM, CLUSTER, ALL_CLASSES) the default is six seconds

Table 10–1 lists the MONITOR PROCESSES class qualifiers.

Table 10–1 MONITOR PROCESSES Class Qualifiers

Description	Qualifier
Top buffered I/O users	/TOPBIO
Top direct I/O users	/TOPDIO
Top CPU users	/TOPCPU
Top page fault users	/TOPFAULT

Example 10–3 illustrates the MONITOR PROCESSES /TOPCPU screen display.

Example 10–3 MONITOR PROCESSES /TOPCPU Screen Display

```
$ MONITOR PROCESSES /TOPCPU

                             VAX/VMS Monitor Utility
                             TOP CPU TIME PROCESSES
                                 on node OTHER
                              27-AUG-1991 17:13:54

                                    0         25        50        75       100
                                    + - - - - + - - - - + - - - - + - - - - -+
 21200524  BATCH_1036            78  *******************************
                                    |         |         |         |          |
 21200646  J_WARTEN               9  ***
                                    |         |         |         |          |
 2120083F  WATTEWS_1              3  *
                                    |         |         |         |          |
                                    |         |         |         |          |
                                    |         |         |         |          |
                                    |         |         |         |          |
                                    |         |         |         |          |
                                    + - - - - + - - - - + - - - - + - - - - -+
```

- The display is updated regularly (every three seconds by default).

Example 10–4 shows the MONITOR screen display of the PAGE class.

Example 10–4 MONITOR Screen Display of the PAGE Class

```
$ MONITOR PAGE

                              VAX/VMS Monitor Utility
                             PAGE MANAGEMENT STATISTICS
                                  on node SPIDER
                               28-AUG-1991 17:04:35
                                        CUR         AVE         MIN         MAX

    Page Fault Rate                   17.41       67.08        0.00     2469.35
    Page Read Rate                     0.32       15.54        0.00      622.68
    Page Read I/O Rate (Hard fault)    0.32        1.05        0.00       12.14
    Page Write Rate                    0.00        0.77        0.00       77.17
    Page Write I/O Rate                0.00        0.00        0.00        0.64

    Free List Fault Rate               7.74        9.19        0.00      212.50
    Modified List Fault Rate           1.29       12.48        0.00     1031.93
    Demand Zero Fault Rate             7.09       34.57        0.00     1269.35
    Global Valid Fault Rate            0.96        9.69        0.00      131.83
    Wrt In Progress Fault Rate         0.00        0.00        0.00        0.00
    System Fault Rate                  0.00        0.00        0.00        0.00

    Free List Size                 61723.00    66910.74    61723.00    68791.00
    Modified List Size              4909.00     4189.02     3855.00     5075.00
```

- Each value is updated regularly (every three seconds by default).
- The current value (CUR) is averaged over the sampling interval (three seconds by default), yielding a rate on a per-second basis.
- Other values are cumulative since the time you entered the MONITOR command.

Example 10–5 MONITOR DECnet

```
$ MONITOR DECNET
                                     VMS Monitor Utility
                                      DECNET STATISTICS
                                        on node LUIGI       From: 26-SEP-1988 10:44:03
                                           SUMMARY          To:   26-SEP-1988 10:44:33

                                        CUR         AVE         MIN         MAX

Arriving Local Packet Rate             0.00        0.33        0.00        0.66
Departing Local Packet Rate            0.99        0.43        0.00        0.99

Arriving Trans Packet Rate             0.00        0.00        0.00        0.00
Trans Congestion Loss Rate             0.00        0.00        0.00        0.00

Receiver Buff Failure Rate             0.00        0.00        0.00        0.00
LRPs Available                        28.00       28.00       24.00       30.00
```

Using MONITOR in a VAXcluster System

MONITOR classes that are cluster-specific:

- CLUSTER gives statistics over the entire cluster

MONITOR qualifiers that are useful in a cluster:

- /NODE gives statistics from particular nodes
- /BY_NODE gives a multinode summary

MONITOR CLUSTER Command

Example 10–6 MONITOR CLUSTER Output

```
$ MONITOR
MONITOR> MONITOR CLUSTER
%MONITOR-I-ESTABCON, establishing connection to remote nodes...

Statistic: CURRENT           VAX/VMS Monitor Utility     27-AUG-1991 16:55:47
                                CLUSTER STATISTICS
                   CPU                    |                MEMORY
                                          |
CPU Busy             0   25   50   75  100|%Memory In Use   0   25   50   75  100
                     +----+----+----+----+|                 +----+----+----+----+
HORSE           100 |*******************|HORSE         94 |******************
TIGER            19 |***                |TIGER         92 |******************
BARNUM           11 |**                 |LION          64 |*************
BAILEY            5 |*                  |BAILEY        43 |*********
RNGLNG            4 |                   |RNGLNG        32 |******
LION              4 |                   |BARNUM        24 |****
BEAR              4 |                   |BEAR          22 |****
------------------------------------------+-------------------------------------
                   DISK                   |                 LOCK
                                          |
I/O Operation Rate   0   25   50   75  100|Tot ENQ/DEQ Rate 0  125  250  375  500
                     +----+----+----+----+|                 +----+----+----+----+
$1$DUA0:          6 |*                  |BARNUM         8 |
$1$DUA0:       R  6 |*                  |RNGLNG           |
$1$DUA1:          2 |                   |TIGER            |
$1$DUA1:       R  1 |                   |HORSE            |
$1$DUA2:            |                   |LION             |
$1$DUA3:            |                   |BAILEY           |
                                          |BEAR             |
```

MONITOR/NODE Qualifier

Example 10–7 Using the /NODE Qualifier with MONITOR DISK

```
MONITOR> MONITOR/NODE=(BARNUM,BAILEY,RNGLNG) DISK
%MONITOR-I-ESTABCON, establishing connection to remote nodes...
                              VAX/VMS Monitor Utility
                         DISK I/O STATISTICS on node BARNUM
                                19-JUN-1991 11:25:11

I/O Operation Rate                          CUR        AVE        MIN        MAX

$1$DUA0:      (CLOWN)  BARNUM_SYS          0.59       6.90       0.00      16.75
$1$DUA1:      (CLOWN)  TIGHTROPE           0.00       0.15       0.00       0.94
$1$DUA2:      (BARNUM) FLYING              0.00       0.00       0.00       0.00
$1$DUA3:      (BAILEY) TRAPEZE             2.09       3.31       0.00      11.59
$2$DUA0:      (HORSE)  THREE               0.00       0.03       0.00       1.21
$2$DUA1:      (HORSE)  RING                0.00       0.00       0.00       0.00

                              VAX/VMS Monitor Utility
                         DISK I/O STATISTICS on node BAILEY
                                19-JUN-1991 11:25:18

I/O Operation Rate                          CUR        AVE        MIN        MAX

$1$DUA0:      (CLOWN)  BARNUM_SYS          0.00       0.16       0.00       1.89
$1$DUA1:      (CLOWN)  TIGHTROPE           0.00       0.05       0.00       0.63
$1$DUA2:      (BARNUM) FLYING              0.00       0.00       0.00       0.00
$1$DUA3:      (BAILEY) TRAPEZE             0.00       0.00       0.00       0.00
$2$DUA0:      (HORSE)  THREE               0.00       0.00       0.00       0.00
$2$DUA1:      (HORSE)  RING                0.00       0.00       0.00       0.00
                             VAX/VMS Monitor Utility
                         DISK I/O STATISTICS on node RNGLNG
                                20-DEC-1989 11:25:16

I/O Operation Rate                          CUR        AVE        MIN        MAX

$1$DUA0:      (CLOWN)  BARNUM_SYS          2.18       1.82       0.00       9.09
$1$DUA1:      (CLOWN)  TIGHTROPE           0.31       0.02       0.00       0.31
$1$DUA2:      (BARNUM) FLYING              0.00       0.63       0.00       7.83
$1$DUA3:      (BAILEY) TRAPEZE             5.00       1.39       0.00       7.82
$2$DUA0:      (HORSE)  THREE               0.00       0.00       0.00       0.00
$2$DUA1:      (HORSE)  RING                0.00       0.00       0.00       0.00
```

An example, Command Procedure for Monitoring the VAXcluster System, exists in the Appendix.

MONITOR Multinode Summary

MONITOR gathers data on only one node.

- You can generate a multinode summary for any MONITOR class. This is especially useful for monitoring total disk activity.

For a side-by-side display of MONITOR data from multiple nodes:

- Use MONITOR to collect data in a file on each node.
- Then use MONITOR/SUMMARY/BY_NODE to combine the data files and produce a multinode summary.

  For example, if BAILEY_MON and BARNUM_MON are the recording files for two nodes:

```
$ MONITOR /NODISPLAY /INPUT=(BAILEY_MON,BARNUM_MON) -
$_ /SUMMARY /BY_NODE DISK
```

Command Procedures

```
$ DIRECTORY SYS$EXAMPLES:*MON*
Directory SYS$COMMON:[SYSHLP.EXAMPLES]
MONITOR.COM;1        MONSUM.COM;1        SUBMON.COM;1
Total of 3 files.
```

- SUBMON.COM
  - Executes the data collection command procedure (MONITOR.COM)
  - Invoked automatically at system startup time when added to SYSTARTUP_V5.COM
- MONITOR.COM collects the data
  - Mails a summary report for each reboot
  - Archives the data collection file
- MONSUM.COM mails two different summary reports
  - One for a 24 hour period
  - One for prime time usage
  - Command procedure resubmits itself (once invoked manually)
- Logicals SYS$MONITOR and MON$ARCHIVE must be defined and these command procedures relocated appropriately
- MONITOR.COM and MONSUM.COM must be edited to specify the user name to mail the reports to
- MONSUM.COM prime time hours may need to be modified for the site
- Edit SYSTARTUP_V5.COM to start up the monitoring command procedure, for example:

  ```
  $ @SYS$MONITOR:SUBMON.COM)
  ```

References

For more information, please refer to the Appendix — Monitoring the System Command Procedures.

MONITOR Command Summary

Table 10–2 lists the MONITOR class names and gives a brief description of each, along with some qualifiers.

Table 10–2 MONITOR Class Names

Class Description	Class Name
All classes	ALL_CLASSES
Brief display of system status in a cluster	CLUSTER
DECnet software statistics	DECNET
Disk I/O statistics	DISK
Distributed lock management statistics	DLOCK
File system statistics	FCP
File system cache statistics	FILE_SYSTEM_CACHE
System I/O statistics	IO
Lock management statistics	LOCK
Time spent in each processor mode	MODES
Disk server statistics in cluster	MSCP
Page management statistics	PAGE
Space allocation in nonpaged dynamic memory	POOL
Statistics on all processes	PROCESSES
RMS file I/O statistics	RMS
System communication services statistics	SCS
Number of processes in each scheduler state	STATES
Brief display of general system status (includes information displayed in other classes)	SYSTEM
Command Qualifier Description	**Command Qualifier Name**
Uses one or more binary input files	/INPUT
Allows turning off screen display	/NODISPLAY
Specifies the nodes for which data is to be collected	/NODE
Generates binary data collection file	/RECORD
Generates summary output file	/SUMMARY

MONITORING WITH DCL COMMANDS

Memory Resources

System performance is strongly dependent on the amount of physical memory.

The **SHOW MEMORY** command displays information about the system's physical memory.

Example 10–8 shows a sample of the SHOW MEMORY command.

Example 10–8 SHOW MEMORY Output

```
$ SHOW MEMORY
              System Memory Resources on 27-AUG-1991 16:39:33.81
Physical Memory Usage (pages):     Total        Free      In Use    Modified
  Main Memory (16.00Mb)         ❶ 32768  ❷ 23954       8516 ❸   298

Slot Usage (slots):                Total        Free    Resident     Swapped
  Process Entry Slots                 30          11          19           0
  Balance Set Slots                   27          10          17           0

Fixed-Size Pool Areas (packets):   Total        Free      In Use        Size
  Small Packet (SRP) List            640         102         538          96
  I/O Request Packet (IRP) List      328          96         232         176
  Large Packet (LRP) List             39          19          20        1648

Dynamic Memory Usage (bytes):      Total        Free      In Use     Largest
  Nonpaged Dynamic Memory         643584       36512      607072       30272
  Paged Dynamic Memory            205312       75600      129712       74480

Paging File Usage (pages):                      Free  Reservable       Total
  DISK$COCOA_SYS:[SYS0.SYSEXE]SWAPFILE.SYS
                                               15000       15000       15000
  DISK$COCOA_SYS:[SYS0.SYSEXE]PAGEFILE.SYS           ❹
                                               23636       -6941       30000

Of the physical pages in use, 3976 pages are permanently allocated to VMS.
```

❶ 32768 pages equates to 16 MB of physical memory

❷ There are 23954 pages currently free for use

❸ Of the 8814 pages in total use, 298 have been modified (written)

❹ Swapping file and paging file free as compared to the total is important

- The negative reservable page count is normal
- VMS algorithm reserves pages for all processes at worst case
- The total pages minus the free pages is what is currently being used
- RSRVPAGCNT system parameter controls this number
- This number would become even more negative if another user logs in

Table 10–3 shows the effect of memory sizes on performance.

Table 10–3 Effect of Memory Sizes on Performance

Item	Description	Problem
Free physical memory	Size of the free page list (the number of pages available for processes that need memory)	If less than a few hundred blocks, swapping occurs.
Free process entry slots	The number of additional processes the VMS system can create	If zero, no users can log in and no new processes can be created.
Free balance set slots	The maximum number of additional processes the VMS system can swap in	If zero, swapping occurs even if there are enough free pages available.
Fixed-size pool areas (packets)	Nonpaged memory in system space, used primarily for I/O	If any item is zero, the system tries to increase it. Enter the SHOW MEMORY /FULL command for more information.
Free paged dynamic memory, free nonpaged dynamic memory	The amount of dynamic memory left for the system to use	If too small, system response deteriorates.
Free swapping file pages	The number of pages available on disk for swapping	If too small, the VMS system uses paging file instead (can significantly reduce performance).
Free paging file pages	The number of pages available on disk for paging	If too small, processes wait in MWAIT state. System prints a message on the console terminal when the paging file reaches 60 percent and 90 percent full.

SHOW SYSTEM Qualifiers

SHOW SYSTEM has two qualifiers, /CLUSTER and /NODE, that are useful in a VAXcluster system.

Example 10–9 SHOW SYSTEM/CLUSTER Command

```
$ SHOW SYSTEM /CLUSTER
VAX/VMS V5.3-2  on node BUZBI  27-AUG-1991 08:56:09.91   Uptime  0 23:54:25
  Pid     Process Name    State  Pri      I/O       CPU       Page flts Ph.Mem
23E00041 SWAPPER          HIB     16        0   0 00:00:06.53         0      0
23E00047 ERRFMT           HIB      7      773   0 00:00:06.39        81    132
23E00048 CACHE_SERVER     HIB     16       74   0 00:00:00.29        66    112
23E00049 CLUSTER_SERVER   CUR      8      446   0 00:00:17.98       128    231
23E0004A OPCOM            HIB      8      278   0 00:00:13.79       275    162
    .
    .
    .
VAX/VMS V5.3-2  on node CANDY  27-AUG-1991 08:56:10.48   Uptime  0 23:46:04
  Pid     Process Name    State  Pri      I/O       CPU       Page flts Ph.Mem
25800041 SWAPPER          HIB     16        0   0 00:00:17.05         0      0
25800047 ERRFMT           HIB      8      775   0 00:00:12.09        81    118
25800048 CACHE_SERVER     HIB     16       84   0 00:00:00.61        62     94
    .
    .
    .
VAX/VMS V5.3-2  on node HARLEY  27-AUG-1991 08:56:11.12   Uptime  0 17:46:28
  Pid     Process Name    State  Pri      I/O       CPU       Page flts Ph.Mem
26E00041 SWAPPER          HIB     16        0   0 00:00:07.67         0      0
26E00084 DECW$WM_1        LEF      7       52   0 00:00:07.84      1286   1683
26E00047 ERRFMT           HIB      8      559   0 00:00:03.51        81    124
26E00048 CACHE_SERVER     HIB     16       12   0 00:00:00.07        62    100
    .
    .
    .
VAX/VMS V5.3-2  on node ZOOZOO  27-AUG-1991 08:56:15.56   Uptime  0 20:37:34
  Pid     Process Name    State  Pri      I/O       CPU       Page flts Ph.Mem
26C00041 SWAPPER          HIB     16        0   0 00:00:08.23         0      0
26C00082 _WSA1:           LEF      6       66   0 00:18:26.22      2681    512
26C00047 ERRFMT           HIB      8      661   0 00:00:04.12        81    132
26C00048 CACHE_SERVER     HIB     16       25   0 00:00:00.14        62    108
26C00049 CLUSTER_SERVER   CUR      8      356   0 00:00:12.63       126    229
    .
    .
    .
26C00050 REMACP           HIB      8        8   0 00:00:00.10        64     49
26C00053 INSPECT$Exec     HIB      8      102   0 00:00:01.73       545     76
26C00054 DTPQUEMAN        LEF      5       36   0 00:00:00.59       257    332
26C00058 DECW$SERVER_0    HIB      6    12593   0 00:08:57.27      9873    970
```

Example 10–10 SHOW SYSTEM/NODE Command

```
$ SHOW SYSTEM /NODE=(HARLEY,WAITER)
VAX/VMS V5.3-2  on node HARLEY  27-AUG-1991 08:57:17.68   Uptime  0 17:47:34
  Pid    Process Name    State  Pri      I/O       CPU       Page flts Ph.Mem
26E00041 SWAPPER         HIB     16        0   0 00:00:07.67         0      0
26E00084 DECW$WM_1       LEF      7       52   0 00:00:07.86      1286   1683
26E00085 MOONSAKA_SM1    LEF      5      419   0 00:00:28.49      9220   2734
26E00047 ERRFMT          HIB      8      559   0 00:00:03.51        81    124
26E00048 CACHE_SERVER    HIB     16       12   0 00:00:00.07        62    100
26E00049 CLUSTER_SERVER  HIB     10      322   0 00:00:08.00       126    221
26E0004A OPCOM           HIB      8      266   0 00:00:10.16       268    128
26E0004B AUDIT_SERVER    HIB     10       42   0 00:00:01.07      1345    221
26E0004C JOB_CONTROL     HIB     10     1493   0 00:00:13.90       415    455
26E0004D CONFIGURE       HIB     10      131   0 00:00:00.47       108    167
26E0004E SMISERVER       HIB      9      581   0 00:00:06.50       399    616
26E0004F NETACP          HIB     10      119   0 00:04:17.39     78235   2048
26E00050 REMACP          HIB      8        8   0 00:00:00.04        64     41
26E00054 DTPQUEMAN       LEF      5       33   0 00:00:00.48       215    390
26E00096 DECW$TE_1       LEF      6     2721   0 00:00:37.31      2395   2420
26E00058 DECW$SERVER_0   HIB      6     5218   0 00:00:59.86      3478   2959
26E00059 MOONSAKA        LEF      4      168   0 00:00:04.80      6174   1298

VAX/VMS V5.3-2  on node WAITER 27-AUG-1991 08:57:18.60   Uptime  8 12:57:26
  Pid    Process Name    State  Pri      I/O       CPU       Page flts Ph.Mem
23800081 SWAPPER         HIB     16        0   0 00:00:34.80         0      0
23800087 ERRFMT          HIB      8    20113   0 00:00:55.02        83    135
23800088 CACHE_SERVER    HIB     16     1383   0 00:00:01.47        63    110
23800089 CLUSTER_SERVER  CUR      9     4303   0 00:01:14.89       135    286
2380008A OPCOM           HIB      9     9233   0 00:01:14.14      2845    287
2380008B AUDIT_SERVER    COM     11     1601   0 00:00:04.76      1379    421
2380008C JOB_CONTROL     HIB      9   347119   0 00:23:36.89       407    599
2380008D CONFIGURE       HIB     10      348   0 00:00:00.34       109    181
2380008E SMISERVER       HIB      9     4566   0 00:00:24.10       493    751
2380008F SYMBIONT_0001   HIB      5     2063   0 00:00:17.52      8606    278
23800090 NETACP          HIB     10   165602   0 01:09:59.83      8984   8869
23800091 REMACP          HIB      8      906   0 00:00:01.10        81     68
23800099 SYMBIONT_0002   HIB      4      362   0 00:00:04.03       801   1161
23802227 YALAYERY        LEF      4     5832   0 00:00:15.13     16280   1234
23801DA8 SERVER_0032     LEF      6     4556   0 00:00:25.49     17685    292   N
23801CAE Judy D          LEF      9     4147   0 00:00:14.25     13018    660
23800AB4 MAIL_16539      LEF      6     1463   0 00:00:07.83      5835    543   N
23801E36 FULTON          LEF      4      743   0 00:00:06.40     10467    266
23800D40 Jack            HIB      7      713   0 00:00:04.26      9588    677
238022C2 Bette           LEF      5    28464   0 00:02:34.32     56859    897
238014C3 BUBBARD         LEF      4     1909   0 00:00:15.36     14190    481
23801EC9 NORM            LEF      4     7820   0 00:00:54.78     56665    426
238015D0 NOTES$000A_1*   HIB      5    24099   0 00:01:03.31     63137    448   N
```

SHOW PROCESS/CONTINUOUS Command

The SHOW PROCESS/CONTINUOUS command displays information about a particular process. The system updates this information every few seconds. To activate the utility, type:

```
$ SHOW PROCESS/CONTINUOUS [/ID=proc-id]  [proc-name]
```

To exit from the utility, type the letter E.

Example 10–11 shows the output from this utility.

NOTE

You must have WORLD privilege to use this command to examine any process on the system whose UIC is not the same as yours.

Example 10–11 Output from SHOW PROCESS/CONTINUOUS

```
$ SHOW PROCESS /CONTINUOUS /ID=7CC

                        Process MAZZE ❶               17:27:29

    State               LEF ❷           Working set            861
    Cur/base priority   9/4 ❸           Virtual pages         3895
    Current PC          7FFEE44C        CPU time       00:10:43.48 ❹
    Current PSL         03C00004        Direct I/O            3691
    Current user SP     7FEF9EBC        Buffered I/O         26600
    PID                 000007CC        Page faults          58835
    UIC                 [GROUP11,MAZZE] Event flags       E0000043
                                                          D4000002
    $1$DUA0:[SYS1.SYSCOMMON.][SYSEXE]MAIL.EXE ❺
```

Notes on Example 10–11:

❶ Name of process

❷ Current state

❸ Current and base priority

❹ CPU time

❺ Name of image being run

USING THE NETWORK CONTROL PROGRAM (NCP) FOR MONITORING

SHOW and LIST Commands

The SHOW command allows the network manager to monitor network activity.

- It provides information from the volatile (memory-resident) network database
  - The volatile database contains current (dynamic) information about network management components
- For example:
  - **Status** information will report current network condition
  - **Characteristics** shows specific static attributes, such as node names and their relevant routing parameters, such as cost
  - **Summary** condenses the status and characteristic information (default display)
  - **Counters** displays error and performance statistics

The LIST command works like the SHOW command, but LIST displays information from the permanent database.

- The permanent (static) database contains values for NCP network parameters.
- It is used at boot time to provide initial values for the volatile network database.

SHOW LINE Command

Lines are the physical data communication paths between nodes. The network manager can use NCP commands to manipulate the physical lines connected to the local node, or to add a line without reconfiguring the node.

Identifying Lines

Every line on a node must have a unique identifier. A line identifier is in the format:

dev-c[-u]

The parts of the line identifier are explained in Table 10–4.

Table 10–4 Line Identification

Part	Function
dev	Mnemonic for a communications interface device, for example: BNA: Ethernet device on VAXBI bus QNA: Ethernet device on Q–bus SVA: Ethernet device on MicroVAX 2000 system UNA: Ethernet device on UNIBUS MFA: FDDI device on XMI DMB: DMB32 synchronous device DMC: DMC11 or DMR11 synchronous device CI: computer interconnect
c	Represents a decimal number (0 or a positive integer) designating the device's hardware controller.
u	Represents a decimal unit or line number (0 or a positive integer) included if the device is a multiple unit line controller.

Example 10–12 Identifying Lines

```
NCP>SHOW KNOWN LINES

Known Line Volatile Summary as of  26-AUG-1991 11:39:09

   Line             State

  QNA-0             on
```

Circuit Commands

Circuits are virtual connections between nodes. The network manager can use NCP commands to manipulate all circuits connected to the local node or to add a circuit without reconfiguring the node.

Identifying Circuits

Each circuit on a node must have a unique identifier. A circuit identifier is in the format:

dev-c or **dev-c-u**

Table 10–5 explains the parts of the circuit identification.

Table 10–5 Circuit Identification

Part	Function
dev	Mnemonic for a communications interface device.
c	Represents a decimal number (0 or a positive integer) designating the hardware controller for the device.
u	Represents a decimal unit or circuit number included only if there is more than one unit associated with the controller.

To display the circuits connected to a node, use the SHOW KNOWN CIRCUITS command, as illustrated in Example 10–13.

Example 10–13 Showing Known Circuits

```
NCP>SHOW KNOWN CIRCUITS

Known Circuit Volatile Summary as of 27-AUG-1991 11:51:20

   Circuit          State                  Loopback     Adjacent
                                             Name      Routing Node

  QNA-0             on                              13.802 (PANHED)
```

SHOW LINKS Command

The following is an extract from NCP>HELP SHOW LINKS

```
Use the SHOW LINKS command  to  display  link  information  (from  the
volatile  database)  available  to the executor node.

SHOW    KNOWN LINKS                     CHARACTERISTICS  TO file-id
        KNOWN LINKS WITH NODE node-id   STATUS
        LINK number                     SUMMARY
```

Example 10–14 Showing Logical Links

```
NCP>SHOW KNOWN LINKS

Known Link Volatile Summary as of 27-AUG-1991 18:19:58

   Link        Node            PID      Process      Remote link  Remote user

  24746  26.974 (HARLEY)   20200053  REMACP              8203   MOONSAKA


NCP>TELL HARLEY SHOW LINK 8203

Link Volatile Summary as of 27-AUG-1991 18:20:07

   Link        Node            PID      Process      Remote link  Remote user

  8203   13.802 (PANHED)   2180011F  _TWA8:             24746   CTERM
```

USING SHOW CLUSTER IN A VAXcluster SYSTEM

SHOW CLUSTER displays a variety of information about the cluster.

There are two types of displays:

- One time display (SHOW CLUSTER)
- Dynamic display (SHOW CLUSTER/CONTINUOUS)

The SHOW CLUSTER display can be modified to include any desired information.

Example 10–15 The Default SHOW CLUSTER Display

```
$ SHOW CLUSTER

     View of Cluster from system ID 1025   node: BARNUM     12-JUN-1991 17:09:35

     +------------------------------+
     |       SYSTEMS       | MEMBERS |
     +------------------------------+
     |  NODE   | SOFTWARE |  STATUS  |
     +------------------------------+
     | BARNUM  | VMS V5.4 | MEMBER   |
     | CLOWN   | HSC V390 |          |
     | BAILEY  | VMS V5.4 | MEMBER   |
     | HIWIRE  | HSC V390 |          |
     +------------------------------+
```

Adding and Removing from the Display

Example 10–16 SHOW CLUSTER Output (Transition Time)

```
Command> REMOVE MEMBERS
Command> ADD TRANSITION_TIME


     View of Cluster from system ID 1126  node: BARNUM   12-JUN-1991 15:31:13

     +--------+-----------------+
     | SYSTEMS|     MEMBERS     |
     +--------+-----------------+
     |  NODE  | TRANSITION_TIME |
     +--------+-----------------+
     | BARNUM | 29-APR-91 21:19 |
     | BAILEY |  3-MAY-91 08:56 |
     | CLOWN  |                 |
     | HIWIRE |                 |
     +--------+-----------------+
```

SHOW CLUSTER Initialization Files

The SHOW CLUSTER utility enables you to specify an initialization file to be used when the utility is invoked.

- Define the logical name SHOW_CLUSTER$INIT to point to this file
- Invoked by the DCL command: SHOW CLUSTER
  - Or use @*filename* from the Command> prompt
- The sample initialization file (SHOW_CLUSTER.COM) shown in Example 10–17 was created with the SAVE command, and illustrates the command format
- Example 10–18 shows the display that is defined by the sample

Example 10–17 A Sample SHOW CLUSTER Initialization File

```
INITIALIZE
REMOVE SYSTEMS
REMOVE SYSTEMS /ID = %X00000000FCD7
REMOVE SYSTEMS /ID = %X000000007311
ADD CL_MEMBERS,LAST_TRANSITION,NODE,HW_TYPE
SET CL_MEMBERS /WIDTH = 6
WRITE CLUSTER2_OUT
```

Example 10–18 Display Resulting from SHOW_CLUSTER$INIT

```
View of Cluster from system ID 69397  node: HANDLE
+-----------------------------------+---------++---------------------------+
|              SYSTEMS              | MEMBERS ||          CLUSTER          |
+--------+--------------------------+---------++---------+-----------------+
|  NODE  |          HW_TYPE         |  STATUS || CL_MEMB | LAST_TRANSITION |
+--------+--------------------------+---------++---------+-----------------+
| HANDLE | VAXstation 3100/GPX      | MEMBER  ||       5 | 25-JUN-91 09:43 |
| JELLY  | VAX 8550                 | MEMBER  |+---------+-----------------+
| WIZARD | VAX 6000-420             | MEMBER  |
| FARMS  | VAX-11/785               | BRK_NON |
+--------+--------------------------+---------+
```

You can control the SHOW CLUSTER display with the commands listed in Table 10–6.

Table 10–6 Basic SHOW CLUSTER Commands

Command	Description
ADD	Add a class or field to the display
REMOVE	Remove a class or field from the display
SET	Change the width or characteristics of a field
INIT	Reset the display to the default state
HELP	Enter interactive help mode
EXIT	Exit the display
SAVE	Create a command procedure SHOW_CLUSTER.COM, which recreates the state of your screen
WRITE	Write current data to a file for problem reports
@filespec	Execute a command procedure of SHOW CLUSTER commands

- SHOW CLUSTER does not automatically prompt for commands.
- Press CTRL/U to see the SHOW CLUSTER prompt (Command>).
- You can also enter a command when the prompt is not visible.

SUMMARY

The VMS operating system provides many methods for examining system activity. Table 10–7 lists some commands and utilities you can use for this purpose.

Some of these commands were covered in other chapters of this course.

Table 10–7 System, Process, and Device Monitoring

Information Displayed	Command or Utility
General System Information	
Overview of the processes on the system	$ SHOW SYSTEM
Overview of print queues	$ SHOW QUEUE/DEVICES/ALL
Overview of batch queues	$ SHOW QUEUE/BATCH/ALL
Overview of mounted disk and tape volumes	$ SHOW DEVICES/MOUNTED
Overview of system memory resources	$ SHOW MEMORY
Demands on system resources	$ MONITOR
Error counts for CPU, memory, and physical devices	$ SHOW ERROR
Cluster activity and performance	$ SHOW CLUSTER
Specific Information (Process or Device)	
Interactive users, terminal names, and process IDs	$ SHOW USERS
Information about current activities of a certain process	$ SHOW PROCESS/CONTINUOUS/ID=pid $ SHOW PROCESS/ALL/ID=pid
Information about user limits and privileges	$ RUN SYS$SYSTEM:AUTHORIZE
Information about disk space allowances	$ SHOW QUOTA /USER=[uic] $ RUN SYS$SYSTEM:SYSMAN
Consumption of resources by processes	$ ACCOUNTING
Information about devices and volumes	$ SHOW DEVICE device

APPENDIX

Monitoring the System Command Procedures

SUBMON.COM executes the data collector command file (MONITOR.COM)

- Logicals SYS$MONITOR and MONITOR$ARCHIVE must be defined in SYS$COMMON:[SYSMGR]SYLOGICALS.COM
- All three command procedures; SUBMON.COM, MONITOR.COM, and MONSUM.COM must be placed in the directory SYS$MONITOR
- A @SUBMON.COM command must be added to SYS$COMMON:[SYSMGR]SYSTARTUP_V5.COM to be invoked at system startup
- MONITOR.COM and MONSUM.COM must be edited
  - Change CLUSTER_MANAGER to your user name or SYSTEM
- MONITOR.COM prime time hours may need to be modified for your site

Example 10–19 SUBMON.COM

```
$  !  Copyright (c) 1987 Digital Equipment Corporation.  All rights reserved.
$  !
$  !  SUBMON.COM  (Submit MONITOR.COM file)
$  !
$  !  This command file is to be placed in a cluster-accessible
$  !  directory called SYS$MONITOR. At system startup time, for
$  !  each node, it is executed by SYSTARTUP.COM, following logical
$  !  name definitions for the cluster-accessible directories,
$  !  SYS$MONITOR and MON$ARCHIVE.
$  !
$  !
$  !  Submit detached MONITOR process to do continuous recording.
$  !
$  !
$ RUN   SYS$SYSTEM:LOGINOUT.EXE -
        /UIC=[1,4]              -
        /INPUT=SYS$MONITOR:MONITOR.COM  -
        /OUTPUT=SYS$MONITOR:MONITOR.LOG -
        /ERROR=SYS$MONITOR:MONITOR.LOG  -
        /PROCESS_NAME="Monitor" -
        /WORKING_SET=100 -
        /MAXIMUM_WORKING_SET=100 -
        /EXTENT=512 -
        /NOSWAPPING
$  !
$  !
$  ! End of SUBMON.COM
$  !
```

Example 10–20 MONITOR.COM

```
$  !  Copyright (c) 1987 Digital Equipment Corporation.  All rights reserved.
$  !
$  !  MONITOR.COM  (Generate MONITOR recording file)
$  !
$  !  This command file is to be placed in a cluster-accessible
$  !  directory called SYS$MONITOR. At system startup time, for each
$  !  node, it creates in SYS$MONITOR a MONITOR recording file, which
$  !  is updated throughout the life of the boot. It also creates in
$  !  MON$ARCHIVE, a summary file from the recording file of the
$  !  previous boot, along with a copy of that recording file.
$  !  Logical name definitions for both cluster-accessible directories,
$  !  SYS$MONITOR and MON$ARCHIVE, must be included in SYSTARTUP.COM.
$  !
$ SET DEF SYS$MONITOR
$ SET NOON
$ PURGE MONITOR.LOG/KEEP:2
$  !
$  !
$  !  Compute executing node name and recording and summary
$  !  file names (incorporating node name and date).
$  !
$ NODE = F$GETSYI("NODENAME")
$ DAY = F$EXTRACT(0,2,F$TIME())
$ IF F$EXTRACT(0,1,DAY) .EQS. " " THEN DAY = F$EXTRACT(1,1,DAY)
$ MONTH = F$EXTRACT(3,3,F$TIME())
$ ARCHFILNAM = "MON$ARCHIVE:"+NODE+"_MON"+DAY+MONTH
$ RECFIL = NODE+"_MON.DAT"
$ SUMFIL = ARCHFILNAM+".SUM"
$  !
$  !
$  !
$  !
$  !  Check for existence of recording file from previous boot
$  !  and skip summary if not present.
$  !
$ OPEN/READ/ERROR=NORECFIL RECORDING 'RECFIL'
$ CLOSE RECORDING
$  !
```

Example 10–20 MONITOR.COM (Cont)

```
$  !
$  !  Generate summary file from previous boot
$  !
$ MONITOR /INPUT='RECFIL' /NODISPLAY /SUMMARY='SUMFIL' -
  ALL_CLASSES,DISK/ITEM=ALL,SCS/ITEM=ALL
$  !
$  !
$  !  Compute subject string and mail summary file to cluster manager
$  !
$ A="""
$ B=" MONITOR Summary "
$ SUB = A+NODE+B+F$TIME()+A
$ MAIL/SUBJECT='SUB' 'SUMFIL' cluster_manager
$  !
$  !
$  !  Archive recording file and delete it from SYS$MONITOR.
$  !
$ COPY 'RECFIL' 'ARCHFILNAM'.DAT
$ DELETE 'RECFIL';*
$  !
$ NORECFIL:
$ SET PROCESS/PRIORITY=15
$  !
$  !
$  ! Begin recording for this boot. The specified /INTERVAL value
$  ! is adequate for long-term summaries; you may require a smaller
$  ! value to get reasonable "semi-live" playback summaries (at the
$  ! expense of more disk space for the recording file).
$  !
$ MONITOR /INTERVAL=600 /NODISPLAY /RECORD='RECFIL' ALL_CLASSES
$  !
$  !
$  ! End of MONITOR.COM
$  !
$  !
```

Example 10–21 MONSUM.COM

```
$  !  Copyright (c) 1987 Digital Equipment Corporation.  All rights reserved.
$  !
$  !  MONSUM.COM  (Generate cluster multi-file summaries)
$  !
$  !  This command file is to be placed in a cluster-accessible directory
$  !  called SYS$MONITOR and executed at the convenience of the cluster
$  !  manager.  The file generates both 24-hour and "prime time" cluster
$  !  multi-file summaries and resubmits itself to run each day at midnight.
$  !
$ SET DEF SYS$MONITOR
$ SET NOON
$  !
$  !  Compute file specification for MONSUM.COM and resubmit the file.
$  !
$ FILE = F$ENVIRONMENT("PROCEDURE")
$ FILE = F$PARSE(FILE,,,"DEVICE")+F$PARSE(FILE,,,"DIRECTORY")+F$PARSE(FILE,,,"NAME")
$ SUBMIT 'FILE' /AFTER=TOMORROW /NOPRINT
$  !
$  !  Generate 24-hour cluster summary.
$  !
$  !
$ MONITOR/INPUT=(SYS$MONITOR:*MON*.DAT;*,MON$ARCHIVE:*MON*.DAT;*) -
   /NODISPLAY/SUMMARY=MONSUM.SUM -
   ALL_CLASSES,DISK/ITEM=ALL,SCS/ITEM=ALL -
   /BEGIN="YESTERDAY+0:0:0.00" /END="TODAY+0:0:0.00" /BY_NODE
$  !
$  !
$  !  Mail 24-hour summary file to cluster manager and delete the file from
$  !  SYS$MONITOR.
$  !
$  !
$  !
$ MAIL/SUBJECT="Daily Monitor Cluster-wide Summary" MONSUM.SUM cluster_manager
$ DELETE MONSUM.SUM;*
$  !
$  !  Generate prime-time cluster summary.
$  !
$  !
$ MONITOR/INPUT=(SYS$MONITOR:*MON*.DAT;*,MON$ARCHIVE:*MON*.DAT;*) -
   /NODISPLAY/SUMMARY=MONSUM.SUM -
   ALL_CLASSES,DISK/ITEM=ALL,SCS/ITEM=ALL -
   /BEGIN="YESTERDAY+9:0:0.00" /END="YESTERDAY+18:0:0.00" /BY_NODE
$  !
$  !
$  !  Mail prime-time summary file to cluster manager and delete the file
$  !  from SYS$MONITOR.
$  !
$  !
$ MAIL/SUBJECT="Prime-Time Monitor Cluster-wide Summary" MONSUM.SUM cluster_manager
$ DELETE MONSUM.SUM;*
$  !
$  !  End of MONSUM.COM
$  !
```

Command Procedure for Monitoring the VAXcluster System

Example 10–22 Command Procedure for MONITOR Recording

```
$! Get the node name to use in the recording file name
$!
$ NODE :== F$GETSYI("NODENAME")
$!
$! Record disk I/O rates for the next hour
$!
$ MONITOR /NODISPLAY /END="+1:00" /RECORD='NODE'.MON DISK
```

Example 10–23 Procedure to Start Recording on Two Nodes

```
$! Record data in batch on node BARNUM
$!
$ SUBMIT /QUEUE=BARNUM_BATCH RECORD.COM
$!
$! Record data in batch on node BAILEY
$!
$ SUBMIT /QUEUE=BAILEY_BATCH RECORD.COM
```

Example 10–24 Two-Node MONITOR Summary

```
$ MONITOR /SUMMARY /BY_NODE /INPUT=(BARNUM.MON,BAILEY.MON) /NODISPLAY DISK
$ TYPE MONITOR.SUM
-------            VMS Monitor Utility
| AVE |            DISK I/O STATISTICS
-------             MULTI-FILE SUMMARY
I/O Operation Rate

Node:      BARNUM            BAILEY
From:27-AUG-1991 15:06 27-AUG-1991 15:06  Row   Row      Row       Row
To:  27-AUG-1991 16:06 27-AUG-1991 16:06  Sum Average  Minimum   Maximum

$1$DUA0:          0.34              0.00  0.3     0.1     0.00      0.34
$1$DUA1:          0.02              1.01  1.0     0.5     0.02      1.01
$1$DUA2:          1.13              3.48  4.6     2.3     1.13      3.48
$1$DUA3:          8.24             12.42 20.6    10.3     8.24     12.42
$2$DUA0:          0.00              0.00  0.0     0.0     0.00      0.00
$2$DUA1:          0.00              0.00  0.0     0.0     0.00      0.00
```

NOTE

This example does not show the entire display because it would require 132 columns.

Developing Command Procedures

INTRODUCTION

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

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

This chapter reviews concepts used in developing command procedures and introduces several advanced features.

OBJECTIVES

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

- Use parameters
- Use terminal I/O to communicate with the user
- Use symbols to manipulate and compare data items
- Control the flow of execution
- Use several commonly used lexical functions to obtain information about the system

RESOURCES

- *Guide to Using VMS Command Procedures*
- *VMS DCL Dictionary*

TOPICS

- Review of command procedure guidelines
- Passing parameters to command procedures
- More on symbols
- Terminal I/O
- More uses for symbols
- Controlling the flow of execution
- Using lexical functions to manipulate data
- Appendix — advanced DCL topics

REVIEW OF COMMAND PROCEDURE GUIDELINES

Steps for Developing Command Procedures

1. **Design the command procedure.**
   - Determine what tasks the procedure should perform.
   - Decide what results the procedure should produce.
2. **Create the command procedure.**
   - Use the text editor of your choice.
   - Specify the file type .COM for the command procedure.
3. **Execute and test the command procedure.**
   - Use the "at" sign (@), followed by the name of the command procedure.

     ```
     $ @LOGIN.COM
     ```
   - Use the DCL command SET VERIFY to:
     - Display each line of the procedure as it executes
     - Help you locate errors if they occur
4. **Modify and retest the command procedure, if necessary.**
   - Repeat steps 2 and 3.
   - Use the DCL command SET NOVERIFY after the procedure has been tested and perfected.
5. **Add comments to the command procedure so it is easy to read and maintain.**
   - Describe the procedure in detail.
   - Describe any parameters that are passed to the procedure.
   - Put lengthy comments either at the end of a procedure (after $EXIT) or at the beginning (skip it with a GOTO) for performance enhancement.

Example 11-1 A Formatted Command Procedure

```
$! D A I L Y . C O M
$!
$! Comment Section
$!
$! DAILY.COM gathers information from the system.  This
$! information is displayed on the terminal or in the
$! DAILY.LOG file if used in batch mode.
$!
$! Author:
$! Modification History:
$!  Person      Date            Change
$!
$ SET NOON
$!
$! Initialization Section
$!
$ SHOW TIME
$!
$! Main Section
$!
$! Capture information on the system
$!
$ SHOW QUOTA
$ SHOW USERS/FULL
$ SHOW PROCESS/ALL
$ DIR/GRAND_TOTAL/SIZE=ALL [...]
$!
$! Find files added and/or changed
$!
$ DIRECTORY/MODIFIED/SINCE=YESTERDAY [...]
$!
$! Cleanup Section
$!
$ EXIT
```

PASSING PARAMETERS TO COMMAND PROCEDURES

You can specify up to eight parameters to a command procedure at execution time.

This allows you to supply the names of files, other procedures, variable values, and other possible values on the command line.

- Format:

```
$ @filename.com  parameter_1  parameter_2  ...  parameter_8
```

- Note that the parameter values are delimited by spaces.
- If you specify parameters when you execute the command procedure, the system assigns the values you specify to the local symbols P1 - P8.

Within the command procedure, *symbol substitution* is the usual method of including the parameter's value in a command line.

— Enclose the symbol name in apostrophes (').

— Within a character string, precede the symbol with two apostrophes ('') and end the symbol with a single apostrophe (').

Example 11–2 Passing a Parameter to a Command Procedure

```
  $ !                                    R E P O R T 2 . C O M
  $
  $   WRITE SYS$OUTPUT ""
  $   WRITE SYS$OUTPUT "Changing your default directory"
  $
  $ ! Set your default to the correct subdirectory
❶ $   SET DEFAULT DISK1:[REPORTS.'P1']
  $
  $   WRITE SYS$OUTPUT ""
  $   WRITE SYS$OUTPUT "Printing the ''P1' report"
  $
  $ ! Print out the report for the correct day
  $   PRINT 'P1'.RPT
  $
  $ ! Return to your login device and directory
  $   WRITE SYS$OUTPUT ""
  $   WRITE SYS$OUTPUT "Changing back to your login directory"
  $
  $   SET DEFAULT SYS$LOGIN
  $   EXIT
  $
  $ ! This command procedure sets your default directory to the [REPORTS.'P1']
  $ ! subdirectory, prints out a report for the day of your choice, returns you
  $ ! to your login device and directory, then exits.
  $
```

Execution of REPORT2.COM:

```
❷ $ @REPORT2 TUESDAY
  Changing your default directory
  Printing the TUESDAY report
  Job TUESDAY (queue SYS$PRINT, entry 47) started on WORK_TXA0
  Changing back to your login directory
```

Notes on Example 11–2:

❶ Force symbol substitution for P1, the first (and only, in this example) parameter passed on the command line.

❷ Here is where the parameter is passed to the command procedure.

Because TUESDAY is the first thing on the command line after the name of the command procedure, it becomes the value of P1.

MORE ON SYMBOLS

As discussed earlier, you can use symbols in many ways in command procedures. DCL translates symbols into their corresponding values.

- Some DCL commands automatically replace symbols with their values.
- Most DCL commands do not perform automatic symbol substitution.
- To force symbol substitution when DCL is not expecting to encounter a symbol:
  - Enclose the symbol name in apostrophes (').
  - In a character string, precede the symbol with two apostrophes ('') and end the symbol with a single apostrophe (').

Table 11–1 Symbol Substitution Techniques

Automatic Substitution	
Command synonym (first item after $ prompt)	`$ XX = "DELETE"` `$ XX FILE.TXT;1`
In the right-hand side of an = or == assignment statement	`$ COUNT = COUNT + 1` `$ FILESPEC = NAME + ".TXT"`
In an IF, WRITE, or INQUIRE command	`$ IF COUNT .GT. 10 THEN -` `WRITE SYS$OUTPUT COUNT`
Forced Substitution	
In a DCL command that does not perform automatic symbol substitution	`$ RUN 'P1'`
In a character string	`$ WRITE SYS$OUTPUT -` `"The file ''P2' exists."`
Concatenating two symbols in a DCL command that does not perform automatic symbol substitution	`$ PRINT 'NAME'.'TYPE'`

Example 11–3 Using Symbol Substitution

```
$ !                                                      REPORT3.COM
$ !
$ !
$ ! This command procedure sets your default directory to the
$ ! [REPORTS.'DAY'] subdirectory, prints out a report for the
$ ! day of your choice, returns you to your login device and
$ ! directory, then exits.
$ !
$!
$   WRITE SYS$OUTPUT ""
$   WRITE SYS$OUTPUT "Day to print report for: ''P1' "
$ !
$   WRITE SYS$OUTPUT ""
$   WRITE SYS$OUTPUT "Changing your default directory"
$ !
$ ! Set your default to the correct subdirectory
$ !
$   SET DEFAULT DISK1:[REPORTS.'P1']
$ !
$   WRITE SYS$OUTPUT ""
$   WRITE SYS$OUTPUT "Printing the ''P1' report"
$ !
$ ! Print out the report for the correct day
$ !
$   PRINT 'P1'.RPT
$ !
$ ! Return to your login device and directory
$ !
$   WRITE SYS$OUTPUT ""
$   WRITE SYS$OUTPUT "Changing back to your login directory"
$ !
$   SET DEFAULT SYS$LOGIN
$   EXIT
```

Execution of REPORT3.COM:

```
$ @REPORT3
Day to print a report: TUESDAY
Day to print report for: TUESDAY
Changing your default directory
Printing the TUESDAY report
Job TUESDAY (queue SYS$PRINT, entry 47) started on WORK_TXA0
Changing back to your login directory
```

TERMINAL I/O

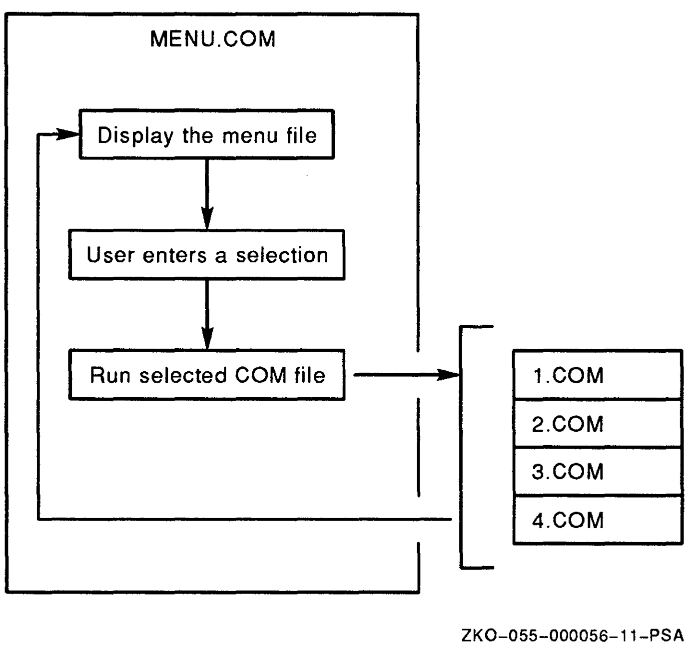
Terminal I/O allows you to write command procedures that interact with the user. You can prompt the user to type information at the keyboard and then use that information in the command procedure. You can process information typed at the keyboard and send it back to the user, for verification, for example.

Terminal input and output can be used to:

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

A simple example of terminal I/O is the creation and use of a menu for a user to select choices from. You can create a menu by displaying formatted text on the screen. Then you can capture the number or letter of a selection displayed as the user types it. Using that selection you can then have the command procedure execute the command procedure that carries out the selected task.

Figure 11–1 How Menu Selection Can be Used to Run Command Procedures



Controlling Terminal I/O

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

In similar fashion, in a command procedure, they can be used to enable communication between the command procedure and the system.

Table 11–2 Logical Names Used with I/O

Logical Name	Description	Interactive	Batch	Command Procedure
SYS$COMMAND	Where the system expects commands to come from	Terminal	Disk where command procedure resides	Terminal
SYS$INPUT	Where the system expects to see input data come from	Terminal	Disk where command procedure resides	Disk where command procedure resides
SYS$OUTPUT	Where the system expects to display the results of any command	Terminal	Disk where log file resides	Terminal
SYS$ERROR	Where the system expects to write error messages	Terminal	Disk where log file resides	Terminal

- You can redefine the values of these logical names to control terminal I/O in a command procedure to perform terminal and file I/O tasks such as:
  - — Sending output to a file instead of to your terminal
  - — Getting a command from a file
  - — Getting input from a data file
- You do not have to open or close files associated with terminal I/O. DCL opens and closes them automatically.

Displaying Information for the User on the Terminal

Use either the WRITE or the TYPE command to display information at the terminal.

The WRITE Command

- Format:

  $ WRITE SYS$OUTPUT   expression

- The **expression** can be:

  — A character string, enclosed in quotation marks

  ```
  $ WRITE SYS$OUTPUT  "Hello"
  ```

  — A symbol name   (The symbol's value is automatically substituted.)

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

  — A lexical function

  ```
  $ WRITE SYS$OUTPUT  F$TIME()
  ```

  — A combination of items separated by commas

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

Example 11–4 Using WRITE SYS$OUTPUT to Display a User Menu

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

Example 11–5 Output from the WRITE SYS$OUTPUT Command Fragment

```
                  SELECT AN ITEM:

      1. ENTER DATA AT THE TERMINAL

      2. ENTER NAME OF DATA FILE
```

Getting Information from the User

Use either the INQUIRE or the READ command to solicit information from the user.

The INQUIRE Command

- Format:

  $ INQUIRE   symbol-name   "prompt"

- Example:

```
$ INQUIRE  NAME  "Enter filename"
```

- The prompt string is optional.
- The user's response is converted to uppercase. Multiple blanks and tabs are replaced with a single space.
- The symbol is equated to the user's response.
- The /NOPUNCTUATION qualifier prevents the display of a colon (:) after the "prompt" string.

The READ Command

- Format:

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

- Example:

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

- The user's response is **not** converted to uppercase, and multiple spaces are **not** removed.
- The symbol is equated to the user's response.

Example 11–6 Terminal I/O in Command Procedures

```
  $!                       I O . C O M
  $!               Demonstration of simple terminal I/O.
  $ TYPE SYS$INPUT
❶ Using INQUIRE

  $ INQUIRE YOU1 "Please type your name"
❷ $ WRITE SYS$OUTPUT YOU1," is using this terminal."
  $ WRITE SYS$OUTPUT ""      ! Blank line.
  $!
  $!
  $ WRITE SYS$OUTPUT "Using READ"
  $ WRITE SYS$OUTPUT ""
  $ READ/PROMPT="Please type your name" SYS$COMMAND YOU2
  $ WRITE SYS$OUTPUT YOU2," is using this terminal."
  $ WRITE SYS$OUTPUT ""
  $ EXIT
```

Sample run of IO.COM:

```
  $ @IO
  Using INQUIRE

❸ Please type your name:  Fred Apon
❹ FRED APON is using this terminal.
❺
  Using READ

❻ Please type your name  Fred Apon
❼ Fred Apon is using this terminal.
  $
```

Notes on Example 11–6:

❶ This is a data line, therefore it does not begin with a dollar sign. The next line, a command line, begins with a dollar sign and terminates the data for TYPE.

❷ The output is a combination of a symbol and some text. Separate the symbol and the text with a comma.

❸ INQUIRE adds a colon and a space to the prompt. (You can suppress the colon with INQUIRE/NOPUNCTUATION.)

❹ INQUIRE converts input to uppercase and reduces multiple spaces to one space.

❺ This blank line is the result of WRITE SYS$OUTPUT "".

❻ Punctuation or spacing is not supplied by READ.

❼ READ preserves the case and spacing of the input.

MORE USES FOR SYMBOLS

Symbols can be used in a variety of ways. The examples listed below briefly show some of these uses. Some of these examples are familiar to you, and some are new.

- **Variables in command procedures**
  - `$ COUNT = 1`
  - If you need to count how many times a task is repeated, you can assign some symbol "COUNT" to one and then increment it by 1 each time the task completes.
- **Reference to data records in a command procedure**
  - `$ READ INPUT_FILE RECORD`
  - Your command procedure would read the contents of INPUT_FILE and store it in the symbol RECORD.
- **Parameters to command procedures**
  - `$ @DOIT.COM DATA.DAT DATA_DONE.DAT`
  - DATA.DAT and DATA_DONE.DAT are values for the symbols P1 and P2, which you can use to pass information to a command procedure on the command line (in this case, the input file and the output file for the procedure).

Symbols Used as Variables in Command Procedures

One of the most common uses for symbols in command procedures is as variables. You can think of variables as *place holders* for values that the command procedure needs to perform a specific task.

For instance, a variable named COUNT could be defined and used to help the command procedure know how many times it has completed a certain task. When you and I do a repetitive task, we just keep *count* in our heads; but a command procedure can't do that. It needs a variable like COUNT that it can keep adding to, to keep track of how many times it has done something.

To use a symbol as a variable:

1. Initialize the symbol

   — Makes sure that no value has been assigned before

   — Defines the symbol for the command procedure

   — `$ COUNT = 0`

2. Use the value of COUNT — Possible uses include:

   — Incrementing COUNT at the end of every loop of a repetitive task

   ```
   $ COUNT = COUNT+1
   ```

   — Using it in the assignment of other symbols

   ```
   $ SUBTOTAL = COUNT+1
   ```

   — Using it in the assignment of string variables

   ```
   $ PARAM := P'COUNT'
   ```

   PARAM now equals P1

   Must use := or :== with strings (local or global)

   Must use apostrophes to force symbol substitution

Using Symbols to Manipulate Data

You can use symbol operations in a number of ways to manipulate data in a command procedure. For example, you could:

- Take a single word from a data record and combine it with a word or words from another data record to create a new record or write a report record
- Remove part of a word or replace parts of words to create a new character string
- Compare strings to test whether the data is what you expected and based on the outcome of that test either do the task or stop the command procedure
- Perform simple arithmetic

These possible tasks may seem rather sterile. However, by applying such simplistic tasks to text in records and files, you can achieve almost any text manipulation task that you need.

Manipulating Strings

You can use these operators to perform string operations:

+ String Concatenation

  Concatenates two character strings to form a single character string.

- String Reduction

  Subtracts one character string from another.

Table 11–3 Examples of String Operations

Example	Result
A = "MYFILE" + ".MEM"	"MYFILE.MEM"
B = "FILENAME.MEM" - "FILE"	"NAME.MEM"
C = "LISTING.LIS" - "LIS"	"TING.LIS"
D = "MIS" + C	"MISTING.LIS"

The following rules apply to string manipulation:

- For string concatenation or reduction to occur, all operands must be character string expressions. Otherwise, any string is converted to an integer and the result is an integer.

  ```
  $ X = "TESTING" + 17
  $ SHOW SYMBOL X
    X = 18   Hex = 00000012  Octal = 00000000022
  ```

  X = 18 because "TESTING" starts with the letter T, which is equivalent to "TRUE", which has a value of 1. 1 + 17 = 18.

- If the string following the minus sign in a string reduction operation occurs more than once in the preceding string, only the first occurrence is removed.

  ```
  $ Y = "TESTING" - "T"
  $ SHOW SYMBOL Y
    Y = "ESTING"
  ```

- If you wish to remove both "T"s:

  ```
  $ Y = "TESTING" -"T" -"T"
  $ SHOW SYMBOL Y
    Y = "ESING"
  ```

Manipulating Arithmetic Expressions

An arithmetic expression can contain:

- Integers
- Lexical functions that evaluate to integers
- Symbols that have integer values
- Integer operands that are connected by arithmetic, logical, and comparison operators

Table 11–4 Arithmetic Operators

Operator	Meaning
+	Arithmetic sum
-	Arithmetic difference
+	Arithmetic unary plus (positive)
-	Arithmetic unary negate (negative)
*	Arithmetic product
/	Arithmetic division (integer quotient)

The following rules apply to arithmetic operations:

- The result of an arithmetic operation is an integer.
- All results giving decimal fractions are truncated.

The truncation of fractional results is shown in Table 11–5.

Table 11–5 Fractional Calculations

Calculation	Actual	Result
99/100	.99	0
101/100	1.01	1
199/100	1.99	1

Using Symbols to Test Data

In addition to being able to manipulate and format data, you can use symbols to test for validity or for certain values to help determine what tasks are performed in a command procedure.

Comparing Strings

All string comparison operators end in the letter S, for "string."

Table 11–6 String Comparison Operators

Operator	Meaning
.EQS.	String equal to
.GES.	String greater than or equal to
.GTS.	String greater than
.LES.	String less than or equal to
.LTS.	String less than
.NES.	String not equal to

The following rules apply to string comparison:

- The comparison is on a character-by-character basis, and terminates when two characters do not match.
- If one string is longer than the other, the shorter string is padded on the right with nulls (a numeric value of %X00) before the comparison is made.

  A null has a lower numeric value than any of the alphabetic or numeric characters.
- Lowercase letters have higher numeric values than uppercase letters.
- If the result of a comparison is true, the expression is given a value of 1.
- If the comparison is false, the expression is given a value of 0.

Table 11–7 Example Character String Comparisons

Comparison Expression	Result	Explanation
"MAYBE" .LTS. "maybe"	1 (true)	The expression is true because the numeric value of "M" is less than that of "m."
"ABCD" .LTS. "EFG"	1 (true)	The expression is true because the numeric value of "A" is less than that of "E."
"YES" .GTS. "YESS"	0 (false)	The expression is false because the numeric value of a null character is less than that of "S". (The "YES" was padded on the right with a null.)
"AAB" .GTS. "AAA"	1 (true)	The expression is true because the numeric value of "B" is greater than that of "A."
"TRUE" .EQS. 1	0 (false)	DCL converts the integer 1 to the string "1" before comparing the numeric value of "T" to the numeric value of "1."
"FALSE" .EQS. 0	0 (false)	DCL converts the integer 0 to the string "0" before making the comparison.
"123" .EQS. 123	1 (true)	DCL converts the integer 123 to the string "123" before making the comparison.

Comparing Arithmetic Expressions

The result of an arithmetic comparison is an integer. Use the arithmetic comparison operators shown in Table 11–8 to compare integer values.

Table 11–8 Arithmetic Comparison Operators

Operator	Meaning
.EQ.	Arithmetic equal to
.GE.	Arithmetic greater than or equal to
.GT.	Arithmetic greater than
.LE.	Arithmetic less than or equal to
.LT.	Arithmetic less than
.NE.	Arithmetic not equal to

The following rules apply to arithmetic comparisons:

- Operands in arithmetic comparisons are integer expressions.
- If you specify a character string value as an operand, the string is converted to an integer value before the comparison is performed.
- If a character string begins with an upper- or lowercase T or Y, the string is converted to the integer 1. If a string begins with any other letter, the string is converted to the integer 0.
- If a string contains numbers that form a valid integer, the string is converted to its integer equivalent.
- If the result of an arithmetic comparison is true, the expression has a value of 1. If the result of the comparison is false, the expression has a value of 0.

Table 11–9 Example Arithmetic Comparisons

Expression	Value of Expression	Explanation
1 .LE. 2	1 (true)	The expression is true because the integer 1 is less than the integer 2.
1 .GT. 2	0 (false)	The expression is false because the integer 1 is not greater than the integer 2.
1 + 3 .EQ. 2 + 5	0 (false)	The expression is false because the integer 4 is not equal to the integer 7.
"TRUE" .EQ. 1	1 (true)	The expression is true because the string "TRUE" is converted to the integer 1 for comparison with the integer 1, and they are equal.
"FALSE" .EQ. 0	1 (true)	The expression is true because the string "FALSE" is converted to the integer 0 for comparison with the integer 0, and they are equal.
"123" .EQ. 123	1 (true)	The expression is true because the string "123" is converted to its integer equivalent for comparison with the integer 123, and they are equal.

CONTROLLING THE FLOW OF EXECUTION

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

As another example, you can write code so that if the user provides the parameters for a command procedure on the command line, they will be used. If not, the command procedure will query the user for the information. This test is accomplished using the DCL commands that control execution flow.

Table 11–10 Commands Used to Control Execution Flow

Command	Function
IF	Tests the value of an expression and executes the specified command or commands if the result of the test is true
GOTO	Transfers control to a different part of the procedure that is identified by a label

IF Command

The IF command accepts multiple statements for execution when the condition is true (as long as the THEN statement is not on the same line as the condition).

- A test of the conditional expression produces the following results:
  - If the condition is *true*, the command(s) following THEN are performed.
  - If the condition is *false*, the next DCL command in sequence is performed or an optional ELSE statement can be performed.
- The command(s) following THEN or ELSE can be any valid DCL command(s).
- There are **three** formats for the IF command.
  - Format 1:

    $ IF conditional-expression THEN command

  - Format 2:

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

  - Format 3:

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

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

Evaluation of Expressions in IF Commands

Expressions are automatically evaluated during the execution of the command.

- Character strings beginning with alphabetic characters that are not enclosed in quotation marks are assumed to be symbol names or lexical functions.
  - The command language interpreter tries to replace these with their current values.
  - The command interpreter does not execute an IF command when it contains an undefined symbol.

    Instead, it issues a warning message and executes the next command in the procedure.
- Symbol substitution in expressions in IF commands is not iterative. Each symbol is replaced only once.
- If you want iterative substitution, precede a symbol name with an apostrophe or an ampersand.

Example 11–7 Conditional Execution

```
.
.
.
$ INQUIRE NCOPY "How many copies do you want?"
$
$ IF NCOPY .LT. 5
$ THEN
        WRITE SYS$OUTPUT "Your copies are printing on the office printer"
        PRINT /QUEUE=OFFICE /COPIES='NCOPY 'P2
$ ELSE
        WRITE SYS$OUTPUT "Your copies are printing on the lab printer"
        PRINT /QUEUE=LAB /COPIES='NCOPY  'P2
$ ENDIF
.
.
.
```

GOTO Command

The GOTO command is used to transfer control to a line that is **not** the next line in the command procedure.

- Format:

  $ GOTO label-name

- Control is transferred to the specified label.
- Do **not** append a colon (:) to the label name in the GOTO statement.

Example 11–8 The GOTO Statement

```
$ GOTO LUNCH           ! Unconditional transfer of control.
$ SHOW USERS           ! We never execute this statement.
$
$ LUNCH:
$   SHOW DEFAULT
$   SHOW TIME
```

Output from this command procedure:

```
$ @GOTO
  WORK3:[SMITH]
  18-MAY-1990 10:50:17
$
```

Iterative Procedures

Use the GOTO command to implement **iteration**, which is the repetition of a group of commands.

To write iterative procedures, use a looping structure.

- Be careful not to create an infinite loop.
- If you accidentally create an infinite loop, press CTRL/Y to stop the procedure. Once the procedure stops, enter the SET VERIFY command and reexecute the procedure to try to find the cause of the problem.

Example 11–9 Using Iteration

```
$ A = 5
$ COUNT = 1
$ LOOP:
$       COUNT = COUNT + 1
$       A = A + 1
$       IF COUNT .LT. 5 THEN GOTO LOOP
$ SHOW SYMBOL A
```

USING LEXICAL FUNCTIONS TO MANIPULATE DATA

Lexical functions can be used to return information about character strings as a symbol value.

You can then manipulate the symbol values in performing tasks.

Lexical functions are much like DCL commands.

- The information provided by a DCL command is usually returned to the terminal.
- Information provided by a lexical function is returned as a symbol value, usable in a command procedure.
- Lexical functions return integer values or character strings depending on their particular function.
- Lexical functions can be used in any context in which you normally use symbols or expressions.

The next few pages introduce some commonly used lexical functions. See the *VMS DCL Dictionary* for details.

F$CVTIME

- Returns information about absolute, combination, or delta time strings.
- Format:

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

- Some of the arguments for this lexical function include:

Output-time-format:

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

Output-time-field:

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

- Example:

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

F$TIME

- Returns the current date and time string
- Format:

```
F$TIME()
```

- The returned string has the following fixed, 23-character format:

    dd-mmm-yyyy hh:mm:ss.cc

- Example:

```
$ NOW = F$TIME()
$
$ SHOW SYMBOL NOW
  NOW = "13-MAY-1989 13:27:15.35"
$
```

F$GETSYI

- Invokes the $GETSYI system service to return status and identification information about your system or a node in your cluster.
- Format:

```
F$GETSYI(item,[node])
```

- Some of the items available for this lexical function:

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

- Example:

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

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

SUMMARY

- Easily read and maintained command procedures follow an organized development cycle that includes:
  - Design
  - Formatting
  - Commenting
  - Testing
- Use terminal I/O to gather and use information typed at the terminal in a command procedure.
  - Use WRITE SYS$OUTPUT to display information to the terminal screen, one line at a time.
  - Use TYPE SYS$INPUT to display information from several lines in the command file on the terminal screen.
  - Use the INQUIRE and READ command to obtain information from the user in command procedures.
- Control the flow of command procedures by using:
  - IF
  - GOTO

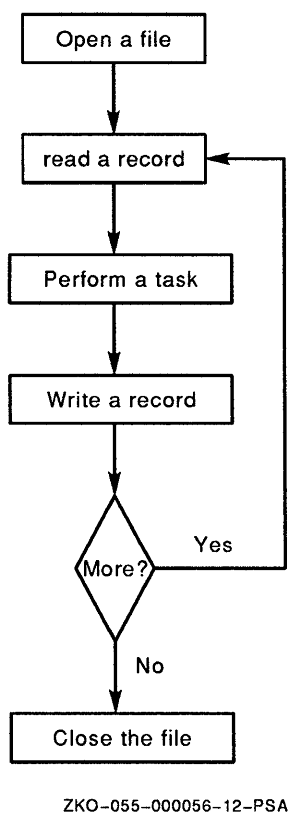
APPENDIX — ADVANCED DCL TOPICS

FILE I/O

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

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

Figure 11–2 File I/O



Opening a File

The OPEN command:

- Assigns a logical name to the file and specifies whether the file is to be read, written, or both.
- Can be used to open a file for read access, for write access, or both.
- Format:

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

- The file-specification should always be the complete form so the user need not be in the directory when running the command procedure.
- Use qualifiers to specify the type of access you desire.

Table 11–11 Qualifiers for the OPEN Command

Qualifier	Description
/APPEND	Opens an existing file and adds records to the end of that file. Cannot be used with the /WRITE qualifier.
/ERROR=label	If an error occurs when opening the file, control is transferred to the specified label.
/READ	Opens an existing file for reading records. (This is the default.) The file must exist.
/WRITE	Opens a file for writing records. Creates a new version of the file if the /READ qualifier is not used.

Closing a File

The CLOSE command closes a file and deassigns the associated logical name.

- Format:

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

Table 11–12 Qualifiers for the CLOSE Command

Qualifier	Description
/ERROR=label	If an error occurs when attempting to close the file, control is transferred to the specified label.
/[NO]LOG	Generates a warning message when you attempt to close a file that was not opened by DCL. (/LOG is the default.)

Example 11–10 Opening and Closing a File

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

Notes on Example 11–10:

❶ Open the file called FILEX.DAT for write access. Assign the logical name OUTPUT to the file. If an error occurs in opening the file, control is transferred to the label ERROR_OPEN.

❷ Close the file, specifying the logical name, not the file specification. Closing a file deassigns the logical name that was created with the OPEN statement.

Reading from a File

The READ command reads a single record from the specified input file and assigns the record's contents to a specified symbol name.

- Format:

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

Table 11–13 Some Qualifiers for the READ Command

Qualifier	Description
/END_OF_FILE=label	This qualifier is used within the context of a loop where control is transferred to the specified label after the last record is read.
/ERROR=label	If an error occurs in reading the file, control is transferred to the specified label.

Example 11–11 The READ Command

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

Notes on Example 11–11:

❶ Open the file for read access.

❷ Read a record from the file. If the end-of-file is encountered, control is transferred to the label ENDIT.

❸ Close the opened file.

❹ Go to this label if an error is encountered while trying to open the file.

Writing to a File

The WRITE command writes the specified data as one record to an open file specified by a logical name.

- Format:

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

Table 11–14 Some Qualifiers for the WRITE Command

Qualifier	Description
/ERROR=label	If an error occurs during a write operation, control is transferred to the specified label.
/SYMBOL	Causes the value of a symbol to be written.
/UPDATE	Replaces the last record read with the record specified in the expression parameter of this WRITE command. (You must be able to read and write to a file to use this qualifier.)

Example 11–12 The WRITE Command

```
❶ $ OPEN/WRITE OUT DATA.NEW
  $ WRITE_LOOP:
❷ $     INQUIRE DATA
  $     IF DATA .EQS. "END" THEN GOTO QUIT
❸ $     WRITE/SYMBOL OUT DATA
  $     GOTO WRITE_LOOP
  $
  $ QUIT:
  $     CLOSE OUT
  $     EXIT
  $
      .
      .
      .
```

Notes on Example 11–12:

❶ Open the file for write access.

❷ Solicit input (DATA) from the terminal.

❸ Write the contents of the symbol DATA to the file.

More Uses for System-Created Logical Names

Earlier, we talked about using SYS$OUTPUT and SYS$INPUT to send and receive information in command procedures. In this section, we are going to talk about redefining these two system logical names to redirect input and output.

Redefining SYS$INPUT

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

For example, to edit a file from a command procedure, include the following lines in the command procedure:

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

- Editors obtain input from SYS$INPUT, which is the command procedure in this case.
- SYS$COMMAND refers to the terminal, the initial input stream when you logged in.
- The /USER_MODE qualifier tells the command procedure that SYS$INPUT is redefined only for the duration of the next image (the editor).
- You can perform edits interactively, then EXIT the editor.
- When the editor image exits, SYS$INPUT resumes its default value, the command procedure file.

Redefining SYS$OUTPUT

Normally, when you issue a command that results in output, (the DIRECTORY command for example), the results are displayed on your screen. When you issue such a command in a command procedure, the default for SYS$OUTPUT remains the screen.

However, you might want to capture that output to do something with it. Do a directory and then edit the information into a report, for example. To do this, you have to redefine SYS$OUTPUT so the results of the command go to a file instead of the screen.

In the following example, the display produced by SHOW DEVICES is directed to MYFILE.LIS in your default directory rather than to your terminal:

```
$ ASSIGN/USER_MODE MYFILE.LIS  SYS$OUTPUT
$ SHOW DEVICES
```

- The /USER_MODE qualifier tells the command procedure that SYS$OUTPUT is redefined only for the duration of the next command.
- Once the image exits, SYS$OUTPUT resumes its default value.

FLOW OF CONTROL

CALL Command

- The CALL command transfers control to a labeled subroutine within a command procedure.
- Format:

  $ CALL label [p1 p2 ... p8]

- **label** indicates the beginning of the subroutine.
- **p1** through **p8** are parameters that can be passed to the subroutine.

SUBROUTINE and ENDSUBROUTINE Commands

- The SUBROUTINE and ENDSUBROUTINE commands define the beginning and end of the subroutine.
- The SUBROUTINE command must be the first executable statement in a subroutine.
- The ENDSUBROUTINE command functions as an EXIT command, if no EXIT command is specified in the procedure.

Example 11–13 Using the CALL Command

```
$ CALL DO_WORK 1 3
$!
$ WRITE SYS$OUTPUT 'SUM'
$!
$ EXIT:
$ EXIT
$!
$ DO_WORK:SUBROUTINE
$!
$ SUM == P1 + P2
$ EXIT
$ ENDSUBROUTINE
```

GOSUB Command

The GOSUB command transfers control to a labeled subroutine in a command procedure without creating a new procedure level.

- Format:

  $ GOSUB label

- **label** indicates the beginning of the subroutine code.
- The GOSUB command does not cause the creation of a new procedure level.
- The RETURN command terminates the GOSUB subroutine procedure, returning control to the command following the calling GOSUB statement.

Example 11–14 Using CALL and GOSUB

```
  $!          A.COM
  $!
❶ $ A = 5
❷ $ B = 5
  $!
❸ $ CALL ADD A B
❽ $ SHOW SYMBOL C
❾ $ EXIT
  $!
❹ $ADD:SUBROUTINE
❺ $ C = P1 + P2
❻ $ EXIT
❼ $ ENDSUBROUTINE
```

```
  $!       B.COM
  $!
❶ $ A = 5
❷ $ B = 5
  $!
❸ $ GOSUB ADD
❼ $ SHOW SYMBOL C
❽ $ EXIT
  $!
❹ $ADD:
❺ $ C = A + B
❻ $ RETURN
```

Notes on Example 11–14:

A.COM

The block of code between SUBROUTINE and ENDSUBROUTINE is never accidentally executed.

❶ Assign the symbol A the value of 5

❷ Assign the symbol B the value of 5

❸ Call a subroutine and pass P1 and P2 values to it

❹ The subroutine creates a new command level

❺ Create a global symbol that adds P1 and P2

❻ Exit (Stores a value in $STATUS)

❼ Mark the end of the subroutine

❽ The value of C has been successfully passed back up a command level because C was defined as a global symbol

❾ Exit A.COM

B.COM

The subroutine can be executed accidentally (protected by EXIT or GOTO.)

❶ Assign the symbol A with the value of 5

❷ Assign the symbol B with the value of 5

❸ Go to the label ADD

❹ Because GOSUB was used, no new command level was created and no parameters are necessary

❺ Create the local symbol C by adding A and B

❻ Return to the next command after the GOSUB command

❼ The value of the symbol C is successfully shown because GOSUB does not create a new command level so the local symbol create in ADD: is available

❽ Exit B.COM

Error Handling

There are two basic ways to handle errors:

- **Reactive**
  - Accept the system default action.
  - Specify your own action using the value of $STATUS.
- **Preventive**
  - Override the default error handling using one of the following:
    - ON WARNING (Catches WARNING, ERROR, SEVERE)
    - ON ERROR (Catches ERROR, SEVERE)
    - ON SEVERE (Catches SEVERE)
    - SET NOON
    - Specify your own action using $STATUS
  - Qualifiers in file I/O statements
    - /END
    - /ERROR
  - Handle interrupts using one of the following:
    - ON CONTROL_Y
    - SET [NO]CONTROL=Y

Status Check

During the execution of a command procedure, the command interpreter checks the condition code returned from each command or program that executes. The condition code can be generated by the system or supplied by the user in the EXIT or RETURN command.

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

Figure 11–3 The Global Symbols $SEVERITY and $STATUS

31 28	27 16	15 03	02 00
CONTROL	FACILITY NUMBER	MESSAGE NUMBER	
$STATUS			$SEVERITY

GSF-RA0294-03-RGS

- Values for $SEVERITY, their meanings and default actions:

0 = Warning (W)	—>	Continue
1 = Success (S)	—>	Continue
2 = Error (E)	—>	Exit
3 = Information (I)	—>	Continue
4 = Severe or fatal error (F)	—>	Exit

- Example:

```
.
.
.
$! Continue if status is S or I. Exit on status E or F.
$! If warning increase previously defined counter, issue message
$!  and continue.
$ RUN TESTPROG.EXE
$ IF $SEVERITY .EQ. 0 THEN GOSUB WARNMSG
$ GOTO REST_OF_PROC
$!
$ WARNMSG:
$ COUNTWARN = COUNTWARN + 1
$ WRITE SYS$OUTPUT "''countwarn' warning(s) have occurred"
$ RETURN
.
.
.
```

ON Command

The ON command specifies an action to be taken when an error condition is encountered in a command procedure.

- Format:

  $ ON condition THEN command

- Temporarily overrides error handling. After execution of the command default error handling is reset, the condition needs to be set again.
- Valid only at the current command level.
- The three forms of the ON command listed are mutually exclusive, since they overlap. Only the last command issued is valid in any sequence.

Table 11–15 Use of the ON Command

	Severity Level of Error and Action Taken		
Command	WARNING	ERROR	SEVERE (FATAL)
By default	CONTINUE	EXIT	EXIT
$ ON WARNING THEN *command*	*command*	*command*	*command*
$ ON ERROR THEN *command*	CONTINUE	*command*	*command*
$ ON SEVERE THEN *command*	CONTINUE	CONTINUE	*command*

Example 11–15 An Example of Using Error Handling — CHOICES.COM

```
  $! CHOICES.COM
  $! This command procedure presents the user with a menu of options.
  $! The user can translate a logical name, find out whether a device
  $! exists on the system, or exit from the command procedure.
  $
  $  SAVE_MESSAGE = F$ENVIRONMENT("MESSAGE")
  $  SET MESSAGE/NOFACILITY/NOSEVERITY/NOIDENTIFICATION/NOTEXT
  $
  $  PRESENT_MENU:
  $  TYPE SYS$INPUT

          Your options are:

          TRANSLATE     Translate a logical name.

          CHECK_DEV     See if a device exists on the system.

          END           Exit from the procedure.

  $  INQUIRE CHOICE "Enter your option"
  $
  $  ON WARNING THEN GOTO ERROR_MESSAGE
❶ $  GOTO 'CHOICE'
  $
❷ $  TRANSLATE:
  $       ON ERROR THEN GOTO END
  $       @COMPROC$NEST:TRANSLATE
  $       GOTO PRESENT_MENU
  $
  $  CHECK_DEV:
  $       ON ERROR THEN GOTO END
  $       @COMPROC$NEST:CHECK_DEV
  $       GOTO PRESENT_MENU
  $
❸ $  ERROR_MESSAGE:
  $       WRITE SYS$OUTPUT " "
  $       WRITE SYS$OUTPUT "Invalid option - please try again"
  $       GOTO PRESENT_MENU
  $
  $  END:
  $       SET MESSAGE 'SAVE_MESSAGE'
  $       EXIT
```

Notes on Example 11–15:

❶ Use a symbol name, rather than specifying a particular label.

❷ Invoke another command procedure if this option is chosen, then go back and refresh the menu.

❸ Print an error message if the user enters an invalid choice, then go back and refresh the menu.

SET NOON Command

The SET NOON command disables any error checking performed by the system.

- The command interpreter continues to place the status code value in $STATUS and the severity level in $SEVERITY, but does not perform any action based on these values.
- Valid at the current command level only.
- Default error handling, or the error handling created by a previous ON command, is restored after EXIT or SET ON.
- Format:

  $ SET [NO]ON

Example 11–16 Using the SET NOON Command

```
.
.
.
$! temporarily disable error check in LOGIN.COM so that
$! if the process name is already set, I don't see an error message
$ SET NOON
$ SET PROCESS/NAME="may be duplicate"
$ SET ON
.
.
.
```

ON CONTROL_Y Interrupts

Use the ON command to handle CTRL/Y interrupts.

- Format:

    $ ON CONTROL_Y THEN command

- $ ON CONTROL_Y remains in effect until:
    - It is changed with another CONTROL_Y command
    - Overruled by SET NOCONTROL=Y command
    - Exit of current command level
- When CTRL/Y is pressed, the current command is aborted and control passes to the specified routine.
- Example:

```
$ ON CONTROL_Y THEN GOTO Y_TRAP
$ ON CONTROL_Y THEN LOGOUT
```

SET NOCONTROL=Y

- Disables CONTROL/Y at all command levels.
- Use with caution.
- Format:

    $ SET [NO]CONTROL=Y

Handling File I/O Errors

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

- Control is passed to the specified label.
- System error message is not displayed.

Example 11–17 Handling File I/O Errors

```
  $ START:
❶ $       OPEN/READ/ERROR=NO_FILE CALLS LOGFILES:CALL.LOG
  $
  $ READ_LOOP:
❷ $      READ/END=DONE CALLS DATA_LINE
  $       WRITE SYS$OUTPUT DATA_LINE
  $       GOTO READ_LOOP
  $!
❸ $ NO_FILE:
  $       WRITE SYS$OUTPUT "Call log doesn't exist."
  $       EXIT
  $
❹ $ DONE:
  $       CLOSE CALLS
  $       EXIT
```

Notes on Example 11–17:

❶ Attempt to open the file, assuming that it exists. If it does not exist, transfer control to the label NO_FILE.

❷ Read a record from the CALLS file. If the end of the file is encountered, transfer control to the label DONE.

❸ The command procedure labled NO_FILE displays a message, and exits. Control is transferred to this label when the OPEN command finds that the file CALLS doesn't exist. Display a message and exit.

❹ When the end-of-file is encountered, control transfers to this label; close the file, and exit.

USING SYMBOLS

Accessing Local and Global Symbol Tables

As you start to use symbols more in command procedures, you will need to understand the difference between local and global symbols in greater detail.

Command Levels

When you log in to a VMS system, you are at what is called DCL command level. Whenever you run a command procedure, you create another command level. If the command procedure that you run, in turn, runs another command procedure, it creates yet another command level.

This structure is not unlike an outline:

DCL command level

  First command procedure command level

    Second command procedure command level

      .
      .
      .

You can create up to 32 of these levels.

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

Local Symbols

- Are available to:
  - The command level that defined them
  - Lower command levels
- Last for the duration of the current command level
- Are created with a single equal sign (=)

```
$ PRINT = "PRINT/NOTIFY"
```

- Are deleted using the following command:

```
$ DELETE/SYMBOL/LOCAL  PRINT
```

(The /LOCAL qualifier is optional, since /LOCAL is the default.)

Global Symbols

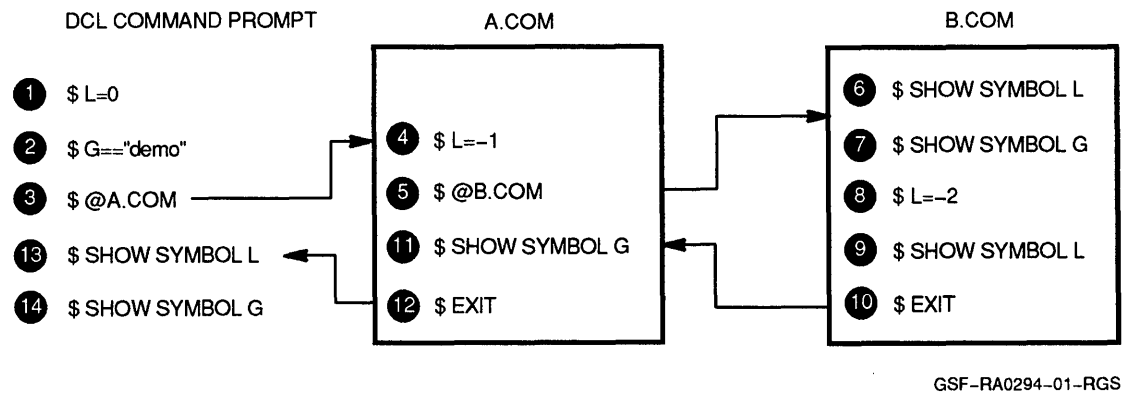
- Are available to all command levels
- Allow you to pass information between command levels
- Are created with a double equal sign (==)

```
$ PRINT == "PRINT/NOTIFY"
```

- Last for the duration of the process
- Are deleted using the following command:

```
$ DELETE/SYMBOL/GLOBAL PRINT
```

Figure 11–4 Accessing Local and Global Symbol Tables



Notes on Figure 11–4:

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

Phases of Command Processing

The command interpreter performs symbol substitution in three phases:

1. **Command Input Scanning**

   a. The command interpreter evaluates symbols preceded by apostrophes **from left to right.**

   b. The command interpreter inserts a null string if a value for the symbol is not found.

2. **Command Parsing**

   a. The command interpreter analyzes the command line and checks the form, function, and syntactical relationship of each part.

   b. Command synonyms are evaluated.

   c. Symbols preceded by an ampersand (&) are evaluated **from left to right.**

   d. The command interpreter inserts a null string if a symbol is not found.

3. **Expression Evaluation**

   a. Replaces symbols not translated until execution time.

   b. Evaluates symbols preceded by the IF and WRITE commands.

   c. Evaluates symbols within lexical functions.

   d. If the symbol is not defined, the command interpreter displays a warning message and stops processing.

Using Symbols to Format Output

Now that you have performed some task on a character string, you want to be able to write the new character string out to a file or record.

Creating Character String Symbols

You have already created character string symbols by using an equal sign and then enclosing the character string in quotes:

```
$ X = "THIS IS A SYMBOL"
```

When using symbol overlays to format data, you need to use a different assignment format. You use:

:= To create a local character string symbol

:== To create a global character string symbol

When creating a symbol with the := or the :==, you do not enclose the character string in quotation marks. The colon-equals tells the VMS operating system that what follows is a character string value.

Example 11–18 Comparing Symbol Creation Methods

```
❶$ X := Richard Mendes
$ SHOW SYMBOL X
  X = "RICHARD MENDES"
$
❷$ Y :== Richard Mendes
$ SHOW SYMBOL Y
  Y = "RICHARD MENDES"
$
❸$ Z = "Richard Mendes"
$ SHOW SYMBOL Z
  Z = "Richard Mendes"
```

Notes on Example 11–18:

❶ Create a local character string symbol. Translation is converted to all uppercase.

❷ Create a global character string symbol. Translation is converted to all uppercase.

❸ Create a literal character string symbol. Case is preserved.

String Overlays

- The format for performing a substring overlay is:

  ```
  $ symbol-name[offset,size] := replacement string
  ```

  (or)

  ```
  $ symbol-name[offset,size] :== replacement-string
  ```

  — *Offset* is an integer that indicates the position of the replacement string relative to the first character in the original string.

  — *Size* is an integer that indicates the length of the replacement string.

- String overlays can be used to create output records with aligned columns.

  — The following example does **not** provide column alignment:

  ```
  $ NAME  := Andrea Vassilos
  $ PHONE := 888-8888
  $ WRITE SYS$OUTPUT NAME,PHONE
  ANDREA VASSILOS888-8888
  ```

  — The following example illustrates how a substring overlay can achieve aligned columns:

  ```
  $ RECORD[0,20] := 'NAME'
  $ RECORD[23,11] := 'PHONE'
  $ WRITE SYS$OUTPUT RECORD
  ANDREA VASSILOS       888-8888
  ```

- If you simply wish to create a blank line, the following overlay example could be used:

```
$ BLANK_LINE[0,80] := ""
```

Arithmetic Overlays

A special form of the assignment statement can be used to perform binary overlays of the current symbol value.

- The format is:

  $ symbol-name[bit-position,size] = replacement-expression
  $ symbol-name[bit-position,size] == replacement-expression

  — **Bit-position** is an integer that indicates the location relative to bit 0, where the overlay is to occur.

  — **Size** is an integer that indicates the number of bits to be overlaid.

- May be used to equate a symbol to ring the bell.

  — The bell is generated by CTRL/G, which has the numeric value of %X07.

  — The following example shows how this would be achieved:

Example 11–19 Ringing the Bell from Inside a Command Procedure

```
❶ $ BELL[0,32]  =  %X07
❷ $ WRITE SYS$OUTPUT BELL
```

Notes on Example 11–19:

❶ The maximum length for bit and size is 32. We are overlaying the entire value.

CTRL/G has a numeric value of %X07. We are associating the value for CTRL/G with the symbol BELL.

❷ Remember that we discussed that the WRITE could be followed by a symbol. Here we are using the symbol BELL to write the value of CTRL/G to SYS$OUTPUT, which is the terminal screen by default. The result is, the bell rings.

LEXICAL FUNCTIONS

Process Information Lexical Functions

Process information lexical functions are used to obtain data about processes. They allow you to save and restore the characteristics of your process.

F$DIRECTORY

- Returns the current default directory as a character string
- Does not return the associated device
- Format:

```
F$DIRECTORY()
```

- Example:

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

F$ENVIRONMENT

- Returns information about the DCL command environment
- Format:

```
F$ENVIRONMENT(item)
```

- Some of the items you can supply to this lexical function:

DEFAULT	Returns the default device and directory name. This information is returned as a character string. The returned string is the same as the output from a SHOW DEFAULT command.
KEY_STATE	Returns a character string indicating the current locked keypad state.
MESSAGE	Returns a character string containing the current setting of the SET MESSAGE command.
PROCEDURE	Returns the file specification for the command procedure from which the F$ENVIRONMENT("PROCEDURE") function is issued. The file specification is returned as a character string. If the lexical function is invoked interactively, a null string ("") is returned.
PROMPT	Returns a character string containing the current prompt string.
PROTECTION	Returns a character string indicating the current default file protection. The string is in a form that can be used with the command SET PROTECTION/DEFAULT to form a valid DCL command line.

- Example:

```
$ CURR_MSG = F$ENVIRONMENT("MESSAGE")
$ SHOW SYMBOL CURR_MSG
  CURR_MSG = "/FACILITY/SEVERITY/IDENTIFICATION/TEXT"
$
$ SET MESSAGE/NOIDENTIFICATION/NOTEXT
$
$ CURR_MSG = F$ENVIRONMENT("MESSAGE")
$ SHOW SYMBOL CURR_MSG
  CURR_MSG = "/FACILITY/SEVERITY/NOIDENTIFICATION/NOTEXT"
$
```

F$GETJPI

- Invokes the $GETJPI system service to return process information about the specified process.
- Format:

```
F$GETJPI(process-id,item)
```

- Some of the items you can supply to this lexical function:

ACCOUNT	Returns the account name containing eight characters with trailing blanks
FILLM	Returns the open file quota (as an integer)
JOBPRCCNT	Returns the number of subprocesses owned (integer)
PID	Returns the process identification number (string)
PRCLM	Returns the subprocess quota (integer)
PRCNAM	Returns the process name (string)
TERMINAL	Returns the login terminal name for interactive users (string)
UIC	Returns the process's UIC (string)
USERNAME	Returns the user name string (12 characters filled with trailing spaces)
WSSIZE	Returns the process's current working set size (integer)

- Example:

```
$ USER = F$GETJPI("","USERNAME")
$ SHOW SYMBOL USER
  USER = "SMITH       "
```

F$MODE

- Returns a character string indicating the mode in which a process is running (for example, INTERACTIVE, NETWORK, BATCH, or OTHER).
- Format:

```
F$MODE()
```

- Example:

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

F$PRIVILEGE

- Returns the value of TRUE or FALSE depending on whether your process has a privilege in the specified list.
- Format:

```
F$PRIVILEGE(priv-list)
```

- Example:

```
$ PRIV = F$PRIVILEGE("BYPASS")
$ SHOW SYMBOL PRIV
  PRIV = "FALSE"
$
$ PRIV = F$PRIVILEGE("TMPMBX")
$ SHOW SYMBOL PRIV
  PRIV = "TRUE"
$
$ SHOW PROCESS/PRIVILEGE

17-MAY-1990 10:38:58.45   User: SMITH          Process ID:   202001C4
                          Node: WHYNOT         Process name: "SMITH"

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

Process rights identifiers:
 INTERACTIVE
 REMOTE
 SYS$NODE_WHYNOT
$
```

F$PROCESS

- Returns the current process name.
- Format:

```
F$PROCESS()
```

- Example:

```
$ PROC_NAME = F$PROCESS()
$ SHOW SYMBOL PROC_NAME
  PROC_NAME = "John Smith"
$
```

F$SETPRV

- Invokes $SETPRV system service to enable or disable specified user privileges.
- Your process must be authorized to set the specified privilege.
- Format:

```
F$SETPRV(priv-states)
```

- Example:

```
$ OLDPRIV = F$SETPRV("OPER,NOTMPMBX")
$ SHOW SYMBOL OLDPRIV

  OLDPRIV = "NOOPER, TMPMBX"
```

F$USER

- Returns the current user identification code (UIC) in alphanumeric format.
- Format:

```
F$USER()
```

- Example:

```
$ UIC = F$USER()
$ SHOW SYMBOL UIC
  UIC = "[GROUP11,SMITH]"
$
```

F$VERIFY

- Returns an integer value indicating whether the procedure verification setting is currently on or off.
- If used with arguments, F$VERIFY can turn the procedure and image verification settings on or off.
- You must include the parentheses after the F$VERIFY function whether or not you specify arguments.
- Format:

```
F$VERIFY([procedure-value][,image-value])
```

- Example:

```
$ SAVE_PROC_VERIFY = F$ENVIRONMENT("VERIFY_PROCEDURE")
$ SAVE_IMAGE_VERIFY = F$ENVIRONMENT("VERIFY_IMAGE")
$ SET NOVERIFY
    .
    .
    .
$ TEMP = F$VERIFY(SAVE_PROC_VERIFY, SAVE_IMAGE_VERIFY)
```

System Information Lexical Functions

System information lexical functions allow the user to obtain information about system parameters such as:

- System message text
- Physical devices
- Logical names and their attributes
- Date and time

F$GETDVI

- Invokes the $GETDVI system service to return information about a physical device.
- Format:

```
F$GETDVI(device-name,item)
```

- device-name can be either a physical or logical name.
- Some of the items available for this lexical function:

DEVCLASS	Number representing device class (disk, tape, etc.)
DEVTYPE	Number representing specific device type
ERRCNT	Number of errors logged on the device
EXISTS	"TRUE" if a device with that name exists
FREEBLOCKS	Number of free blocks on a disk volume
MNT	"TRUE" if device is mounted

- Example:

```
$ MOUNTED = F$GETDVI("WORK12:","MNT")
$ SH SYM MOUNTED
  MOUNTED = "TRUE"
$
```

F$MESSAGE

- Returns the message text associated with a specific system status code
- Format:

```
F$MESSAGE(status-code)
```

- Example:

```
$ RUBBISH
%DCL-W-IVVERB, unrecognized command verb --- check validity and spelling
 \RUBBISH\
$
$ SHOW SYMBOL $STATUS
  $STATUS = "%X00038090"
$
$ ERROR = F$MESSAGE($STATUS)
$ SHOW SYMBOL ERROR
  ERROR = "%CLI-W-IVVERB, unrecognized command verb --- check validity and
spelling"
$
```

F$TIME

- Returns the current date and time string
- Format:

```
F$TIME()
```

- The returned string has the following fixed, 23-character format:

  dd-mmm-yyyy hh:mm:ss.cc

- Example:

```
$ NOW = F$TIME()
$
$ SHOW SYMBOL NOW
  NOW = "13-MAY-1989 13:27:15.35"
$
```

F$TRNLNM

- Returns the equivalence string or requested attributes associated with the logical name.
- Does not perform iterative translation.
- Format:

```
F$TRNLNM(logical-name[,table][,index][,mode][,case][,item])
```

- Some of the values you can supply for the *item* argument:

CONCEALED	Returns one of the character strings "TRUE" or "FALSE" to indicate whether the CONCEALED attribute was specified with the /TRANSLATION_ATTRIBUTES qualifier when the logical name was created. The CONCEALED attribute is used to create a concealed logical name.
CONFINE	Returns one of the character strings "TRUE" or "FALSE" to indicate whether the logical name is confined. If the logical name is confined (true), then the name is not copied to subprocesses. If the logical name is not confined (false), then the name is copied to subprocesses.
TABLE	Returns one of the character strings "TRUE" or "FALSE" to indicate whether the logical name is the name of a logical name table.
TABLE_NAME	Returns the name of the table where the logical name was found.
TERMINAL	Returns one of the character strings "TRUE" or "FALSE" to indicate whether the TERMINAL attribute was specified with the /TRANSLATION_ATTRIBUTES qualifier when the logical name was created.
VALUE	Returns the equivalence name associated with the specified logical name. If the logical name has more than one equivalence name, the F$TRNLNM function returns the name specified by the index argument. VALUE is the default if you do not specify an item argument.

- Example:

```
$ ASSIGN/TRANSLATION_ATTRIBUTES=TERMINAL TESTDATA.COM TEST
$
$ SHOW LOGICAL/FULL TEST
   "TEST" [super] = "TESTDATA.COM" [terminal] (LNM$PROCESS_TABLE)
$
$ X = F$TRNLNM("TEST",,,,,"TERMINAL")
$
$ SHOW SYMBOL X
  X = "TRUE"
$
```

Character Manipulation Lexical Functions

Character manipulation lexical functions allow you to obtain and operate on character strings. Some of the operations you can do are:

- Return character strings to you in a specified format
- Extract specific parts of a character string
- Edit a string that is returned to you
- Convert character and numeric input to character strings
- Return the integer equivalent of a specified string
- Return the length of a string
- Locate a substring within a string and return the offset position

F$EDIT

- Edits a string expression
- Format:

```
F$EDIT(string,edit-list)
```

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

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

- Example:

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

F$EXTRACT

- Extracts a substring from a character string expression.
- Format:

```
F$EXTRACT(offset,length,string)
```

- Example:

```
$ FULL_NAME = "FRED BLOGGS"
$ FIRST_NAME = F$EXTRACT(0,4,FULL_NAME)
$ SHOW SYMBOL FIRST_NAME
  FIRST_NAME = "FRED"
$
```

F$FAO

- Converts character and numeric input to character strings. (FAO stands for Formatted ASCII Output).
- Format:

```
F$FAO(control string[,arg1,arg2...arg15])
```

- Example:

```
$ A = "ERR"
$ B = "IS"
$ C = "HUMAN"
$ PHRASE = F$FAO("TO !#(#AS)", 3, 6, A, B, C)
$ SHOW SYMBOL PHRASE
  PHRASE = "TO ERR   IS    HUMAN "
```

F$INTEGER

- Returns an integer equivalent of the specified string.
- Format:

```
F$INTEGER(string)
```

- Example:

```
$ A = F$INTEGER("TRUE")
$ SHOW SYMBOL A
  A = 1   Hex = 00000001  Octal = 00000000001
$
$ A = F$INTEGER("FALSE")
$ SHOW SYMBOL A
  A = 0   Hex = 00000000  Octal = 00000000000
$
```

F$LENGTH

- Returns the length of a string.
- Format:

```
F$LENGTH(string)
```

- Example:

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

F$LOCATE

- Locates the substring in the string and returns the offset position.
- Format:

```
F$LOCATE(substring,string)
```

- Example:

```
$ LONG_STRING = "This is a long string. We are looking for the word 'long'"
$ FIND = F$LOCATE("long",LONG_STRING)
$ SHOW SYMBOL FIND
  FIND = 10   Hex = 0000000A  Octal = 00000000012
$
```

File Information Lexical Functions

File information lexical functions allow you to:

- Return attribute information for a specific file, in integer or character form
- Parse a file specification and return the whole or any part of it
- Search a directory file and return a file specification

F$FILE_ATTRIBUTES

- Returns attribute information for a specific file, in integer or character string form, depending upon the item you request.
- Format:

```
F$FILE_ATTRIBUTES(file-spec,item)
```

- Some of the values you can supply for *item*:

ALQ	Allocation quantity (as an integer)
BDT	Backup date/time string
CDT	Creation date/time string
CTG	True, if contiguous Returns the value "TRUE" or "FALSE"
EDT	Expiration date/time string
EOF	Number of blocks used (integer)
FID	File ID string
KNOWN	Known file Returns the value "TRUE" or "FALSE" to indicate whether file is installed with the Install utility
ORG	File organization Returns the value "SEQ", "REL", or "IDX"
PRO	File protection string
UIC	Owner UIC string

```
$ DIRECTORY/PROTECTION/OWNER

Directory SYS$SYSDEVICE:[SMITH]

TEST.COM;5          [SMITH]          (RWED,RWED,RE,)
TEST.DIR;1          [SMITH]          (RWE,RWE,RE,E)

Total of 2 files.
$
$ PROT = F$FILE_ATTRIBUTES("TEST.COM","PRO")
$ SHOW SYMBOL PROT
   PROT = "SYSTEM=RWED, OWNER=RWED, GROUP=RE, WORLD"
$
$ OWNER = F$FILE_ATTRIBUTES("TEST.COM","UIC")
$ SHOW SYMBOL OWNER
  OWNER = "[SMITH]"
$
```

F$PARSE

- Invokes the $PARSE RMS routine to parse a file specification and to return either the expanded file specification or the particular file specification field you request.

- Format:

```
F$PARSE(file-spec[,default-spec][,related-spec][,field][,parse-type])
```

- Values you can supply for the *field* argument:

NODE	Node name
DEVICE	Device name
DIRECTORY	Directory name
NAME	File name
TYPE	File type
VERSION	File version number

- Values you can supply for the *parse-type* argument:

NO_CONCEAL	Logical names are not concealed. Therefore, logical name translation does not end when a concealed logical name is encountered.
SYNTAX_ONLY	The syntax of the file specification is checked without verifying that the specified directory exists on the specified device.

- Example:

```
$ FULL_FILE_SPEC = F$PARSE("TEST.COM")
$ SHOW SYMBOL FULL_FILE_SPEC
  FULL_FILE_SPEC = "SYS$SYSDEVICE:[SMITH]TEST.COM;"
$
$ FULL_FILE_SPEC = F$PARSE("TEST.COM",,,version,"NO_CONCEAL")
$ SHOW SYMBOL FULL_FILE_SPEC
  FULL_FILE_SPEC = "WHYNOT$DUA0:[SMITH]TEST.COM;"
$
```

F$SEARCH

- Invokes the $SEARCH RMS routine to search a directory file and to return the full file specification of a file you name.

- Format:

```
F$SEARCH(file-spec[,stream-id])
```

- Example:

```
$ IS_IT_THERE = F$SEARCH("TEST.COM")
$ SHOW SYMBOL IS_IT_THERE
  IS_IT_THERE = "SYS$SYSDEVICE:[SMITH]TEST.COM;5"
$
$ IS_IT_THERE = F$SEARCH("NONEXISTENT.FILE")
$ SHOW SYMBOL IS_IT_THERE
  IS_IT_THERE = ""
$
```

- *Stream-id*

  — The search stream identification number is used to maintain separate search contexts when you use the F$SEARCH function more than once and when you supply different file-spec arguments.

  — If you use F$SEARCH more than once in a command procedure, and if you also use different file-spec arguments, specify stream-id arguments to identify each search separately.

  — If you omit the stream-id, F$SEARCH assumes an implicit single search stream. That is, it starts searching at the beginning of the directory file each time you specify a different file-spec argument.

Written Exercises

QUEUE MANAGEMENT

Written Exercise 3-1 — Queues

1. What is the name of the system process that schedules print and batch jobs?
2. What is the name of the system process that performs many other system management and control tasks?
3. Match the type of queue with its function.

Queue Type	**Action**
___Moves jobs to specified execution queues when resources for executing the job are available	a. Execution b. Generic
___Prints the file or processes the batch job	

Solutions to Written Exercise 3-1 — Queues

1. What is the name of the process that schedules print and batch jobs?

   `QUEUE_MANAGER`

2. What is the name of the system process that performs many other system management and control tasks?

   `JOB_CONTROL`

3. Match the type of queue with its function.

Queue Type	Action
_b_ Moves jobs to specified execution queues when resources for executing the job are available	a. Execution
_a_ Prints the file or processes the batch job	b. Generic

Written Exercise 3-2 — Queue Creation

1. List the five steps taken to establish a print execution queue:

   a.

   b.

   c.

   d.

   e.

2. Write the DCL command that you would use to display all device queues.

3. Write the DCL command that you would use to create and start a print queue.

Solutions to Written Exercise 3-2 — Queue Creation

1. List the five steps taken to establish a print execution queue:

   a. Set physical attributes of device

   b. Spool each printing device

   c. Initialize and start an execution queue for each printing device

   d. Initialize and start a generic print queue

   e. Initialize and start a generic terminal queue

2. Write the DCL command that you would use to display all device queues.

   ```
   $ SHOW QUEUE/DEVICE
   ```

3. Write the DCL command that you would use to create and start a print queue.

   ```
   $ INITIALIZE/QUEUE/START/ON=LPA0 SYS$PRINT
   ```

MAINTAINING SYSTEM SECURITY

Written Exercise 8-1 — Process Parameters

Example 12–1 displays the characteristics of a process that has several 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)

Example 12–1 Process Parameters of a Sample Interactive Process

```
$ SHOW PROCESS/ALL
19-APR-1990 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: DISK$COMPACT:[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:

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

Solutions to Written Exercise 8-1 — Process Parameters

Answers	Parameters
MUSIC	Account name
DISK$COMPACT:[GABRIEL]	Default device and directory specification
LTA28:	Interactive terminal specification
20200242	Process identification number
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
CMRKNL, TMPMBX, OPER, NETMBX	Privileges (list them)

Laboratory Exercises

INTRODUCTION

The exercises in this chapter are organized to practice skills taught in each chapter. You will find exercises pertaining to each chapter under the chapter title as a heading in this chapter.

More importantly, the exercises in this course are designed as several exercises with many parts. Each chapter-associated exercise practices a particular skill or skills learned in that chapter.

Each laboratory section is followed by a matching section that discusses output you may have seen, gives hints, or actually provides sample code that would accomplish the tasks assigned in the laboratory exercise.

The solution that is included after each exercise identifies just one way to approach the exercise. As is always the case with any exercise, there may be several different approaches that are just as valid.

The exercises all assume that you have a student account with privileges of a system manager, and that you have already logged in to the system.

LABORATORY EXERCISES — MANAGING DISKS

Laboratory Exercise 1-1 — Devices

Perform the following exercises at an interactive terminal.

1. Enter the command to list devices on the system.
2. Enter the command to list the complete status of devices on the system.
3. Enter the command to list only disks on the system.
4. Enter the command to list files that are currently open.

Solutions to Laboratory Exercise 1-1 — Devices

1. Enter the command to list devices on the system.

   ```
   $ SHOW DEVICE
   ```

2. Enter the command to list the complete status of devices on the system.

   ```
   $ SHOW DEVICE/FULL
   ```

3. Enter the command to list only disks on the system.

   ```
   $ SHOW DEVICE D
   ```

4. Enter the command to list files that are currently open.

   ```
   $ SHOW DEVICE/FILES
   ```

Laboratory Exercise 1-2 — Devices

At an interactive terminal, enter the command to produce the following information on your device:

1. What type of disk is it?
2. Is it mounted on the local system?
3. Is it mounted on any other system in the cluster?
4. How big is it? (in 512-byte blocks)
5. How many blocks are free?
6. What is the owner UIC and volume protection?
7. Has it generated any hardware errors since the system was started up?

Solution to Laboratory Exercise 1-2 — Devices

The command used to produce this disk information is:

```
$ SHOW DEVICE/FULL $1$DUA0
```

Here is a sample output from that command:

```
$ SHOW DEVICE/FULL $1$DUA0

Disk $1$DUA0:(BARNUM), device type RA81,(1) is online, mounted,
    (2)file-oriented device,
    shareable, served to cluster via MSCP Server, error logging is enabled.

    Error count                    0(7) Operations completed             5989
    Owner process                 ""    Owner UIC                      [1,1](6)
    Owner process ID        00000000    Dev Prot     S:RWED,O:RWED,G:RWED,W:RWED(6)
    Reference count                1    Default buffer size               512
    Total blocks              891072(4) Sectors per track                  51
    Total cylinders             1248    Tracks per cylinder                 8
    Allocation class               1

    Volume label            "FLYING"    Relative volume number              0
    Cluster size                   3    Transaction count                  93
    Free blocks                 8069(5) Maximum files allowed          222768
    Extend quantity                5    Mount count                         7
    Mount status              System    Cache name           "_$1$DUA0:XQPCACHE"
    Extent cache size             64    Maximum blocks in extent cache   2088
    File ID cache size            64    Blocks currently in extent cache    0
    Quota cache size              30    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 RNGLNG, BAILEY, LION, HORSE, BEAR, TIGER.(3)
```

This output answers questions including the following, which are keyed to the example.

1. What type of disk is it?
2. Is it mounted on the local system?
3. Is it mounted on any other system in the cluster?
4. How big is it? (in 512-byte blocks)
5. How many blocks are free?
6. What is the owner UIC and volume protection?
7. Has it generated any hardware errors since the system was started up?

Laboratory Exercise 1-3 — Disk Quotas

NOTE: You cannot perform these exercises unless you have write access to the quota file on your class volume. Check with your instructor 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.

Solutions to Laboratory Exercise 1-3 — Disk Quotas

1. No solution needed.

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], 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 /DEVICE=CLASS_DISK [320,10]`

   Substitute the name of your class volume for CLASS_DISK, and your own UIC for [320,10]. The utility modifies the quota file dynamically, so it recorded the blocks you used for the file when you created it.

5. `SYSMAN> DISKQUOTA MODIFY [320,10] /PERMQUOTA=2000 /OVERDRAFT=300`

   Substitute the appropriate values that correspond to your record.

6. `SYSMAN>EXIT`

7. `$ SET DEFAULT SYS$SYSTEM:`

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

   Substitute the name of your class volume for CLASS_DISK and your own UIC for [320,10]. If you do not enter the /DEVICE qualifier, the utility uses the quota file for the current default disk (the system disk), if it exists. That file does not contain your quota record.

LABORATORY EXERCISES — USING LOGICAL NAME TABLES

Laboratory Exercise 2-1 — Logical Name Tables

1. Use the DCL command SHOW LOGICAL to find out the physical device name of your default login device.
2. Find out the physical device name of the default login device for one of your fellow students, given his or her user name.
3. Assign an additional logical name to the device whose name you found in question 1:
   - Make your user name part of the logical name. For example, if your user name is LAB10, you could call your logical name LAB10_DISK.
   - Assign it in the system table.
   - Give it the attributes CONCEALED and TERMINAL.

   Use the name in one or more DCL commands to verify that it works.
4. Use SYSMAN to make the same logical name assignment clusterwide.

   If possible, log in to another node of the cluster and verify that the name is defined.

   Finally, use SYSMAN again to deassign your logical name clusterwide.
5. Find out whether the logical name SYS$SYSROOT has any translation attributes defined.
6. In the command procedure, SYS$STARTUP:SYLOGICALS.COM, look for the ASSIGN or DEFINE commands. (Hint: use the DCL SEARCH command.)

   For each logical name you see assigned in this procedure, use the SHOW LOGICAL or SHOW TRANSLATION command to verify that it is defined on the system.

Solutions to Laboratory Exercise 2-1 — Logical Name Tables

1. In the following example, 1DUA12 is the physical device name.

```
$ SHOW LOGICAL SYS$LOGIN
  "SYS$LOGIN" = "USERDISK1:[MATTHEWS]" (LNM$JOB_807262A0)
$ SHOW LOGICAL USERDISK1
  "USERDISK1" = "$1$DUA12:" (LNM$SYSTEM_TABLE)
```

   Instead of SYS$LOGIN, you could have used the logical name SYS$LOGIN_DEVICE or SYS$SCRATCH. You could use SYS$DISK if you have not used SET DEFAULT to change your default device.

2. In the following example, 100DUA1 is the physical device name.

```
$ RUN SYS$SYSTEM:AUTHORIZE
UAF> SHOW STUDENT9

Username: STUDENT9                          Owner:  System & Network Mgt II
Account:  LAB                               UIC:    [22,11] ([LAB,STUDENT9])
CLI:      DCL                               Tables:
Default:  LABDISK:[STUDENT9]
    .
    .
    .
UAF> CTRL/Z
$ SHOW LOGICAL LABDISK
  "LABDISK" = "$100$DUA1:" (LNM$SYSTEM_TABLE)
```

3. In the following example, 1DUA12 is the physical device name.

```
$ ASSIGN /SYSTEM /TRANSLATION_ATTRIBUTES=(CONCEALED,TERMINAL) -
$_ $1$DUA12: LAB10_DISK
$ DIRECTORY LAB10_DISK:[000000]

Directory LAB10_DISK:[000000]

000000.DIR;1        BACKUP.SYS;1        BADBLK.SYS;1        BADLOG.SYS
BADLOG.SYS;1        BITMAP.SYS;1        CONTIN.SYS;1        CORIMG.SYS;1
INDEXF.SYS;1        LAB1.DIR;1          LAB2.DIR;1          LAB3.DIR;1
    .
    .
    .
```

4.

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SET ENVIRONMENT/CLUSTER
%SYSMAN-I-ENV, current command environment:
        Clusterwide on local cluster
        Username LAB10 will be used on nonlocal nodes

SYSMAN> DO ASSIGN /SYSTEM /TRANSLATION=CONCEALED $1$DUA12: LAB10_DISK
%SYSMAN-I-OUTPUT, command execution on node BARNUM
%DCL-I-SUPERSEDE, previous value of LAB10_DISK has been superseded
%SYSMAN-I-OUTPUT, command execution on node NODE2
    .
    .
    .
SYSMAN> CTRL/Z
$ SET HOST NODE2
    .
    .
    .
$ SHOW LOGICAL LAB10_DISK
  "LAB10_DISK" = "$1$DUA12:" (LNM$SYSTEM_TABLE)
$ LOGOUT
  Process LAB10 logged out at  5-SEP-1991 17:39:21.66
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SET ENVIRONMENT/CLUSTER
%SYSMAN-I-ENV, current command environment:
        Clusterwide on local cluster
        Username LAB10 will be used on nonlocal nodes

SYSMAN> DO DEASSIGN /SYSTEM LAB10_DISK
%SYSMAN-I-OUTPUT, command execution on node BARNUM
%SYSMAN-I-OUTPUT, command execution on node NODE2
    .
    .
    .
SYSMAN> CTRL/Z
```

5. The logical name has the attributes CONCEALED and TERMINAL. (So does one of its equivalence names, SYS$COMMON.)

```
$ SHOW LOGICAL SYS$SYSROOT /FULL
   "SYS$SYSROOT" [exec] = "$1$DUA0:[SYS11.]" [concealed,terminal] (LNM$SYSTEM_TA
BLE)
        = "SYS$COMMON:"
1  "SYS$COMMON" [exec] = "$1$DUA0:[SYS11.SYSCOMMON.]" [concealed,terminal] (LNM$
SYSTEM_TABLE)
```

6. `$ SEARCH SYS$STARTUP:SYLOGICALS.COM /MATCH=OR DEFINE,ASSIGN`

System logical names vary from system to system. See your instructor if you need help with this exercise.

LABORATORY EXERCISES — QUEUE MANAGEMENT

Laboratory Exercise 3-1 — Queues

The following exercises allow you to practice DCL commands needed to create, manipulate, view, and delete queues.

Perform the following laboratory exercises at an interactive terminal.

1. Issue a command that lists the allocated devices.

2. Use the command procedure LOGIN.COM in your main directory as a sample to place in the print and batch queues. To place several entries in the print queues, issue the following commands:

   ```
   $ PRINT/HOLD LOGIN.COM
   $ PRINT/HOLD/QUEUE=LPA0  LOGIN.COM
   ```

   Examine the print queues with the following commands:

   ```
   $ SHOW QUEUE queue_name
   $ SHOW QUEUE/DEVICES/ALL
   ```

3. Place several copies of the LOGIN.COM file into the batch queues by issuing the following commands:

   ```
   $ SUBMIT/HOLD LOGIN.COM
   $ SUBMIT/HOLD/PRIORITY=1 LOGIN.COM
   $ SUBMIT/HOLD/PRINT=LPA0: LOGIN.COM
   ```

   NOTE

   These files are not activated in the batch queue because of the the HOLD qualifier. Again, using the HOLD qualifier allows you to examine the batch jobs with the correct DCL commands.

   As before, the system returns a job number for each SUBMIT command you issue. Record these numbers for later use.

4. Examine the batch queues with the following commands:

   ```
   $ SHOW QUEUE/BATCH
   $ SHOW QUEUE/BATCH/ALL
   ```

   Notice the different information obtained from each batch queue command. Examine the information given with each job.

5. Using the job entry numbers for each print and batch job, delete those jobs from the queues using the command:

```
$ DELETE/ENTRY=job_entry_number queue_name
```

For example, if the system returns:

```
JOB SYSTEM (queue SYS$PRINT, entry 390) holding
```

delete the job with the DCL command:

```
$ DELETE/ENTRY=390 SYS$PRINT
```

Using the job entry numbers you recorded, delete any remaining jobs from the queue. If problems arise, use the HELP command (for example, HELP DELETE/ENTRY) and the *VMS DCL Dictionary* to get more information. If you still have trouble deleting the entries, contact the instructor.

Solutions to Laboratory Exercise 3-1 — Queues

1. Issue a command that lists the allocated devices.

```
$ SHOW DEVICES/FULL
```

2. No solution needed.
3. No solution needed.
4. No solution needed.
5. No solution needed.

Laboratory Exercise 3-2 — Batch Queues

Preparation for this lab exercise:

Set your default to your main directory.

Use an editor or the CREATE command to create a command procedure that you can use in the following exercise. Name this file B.COM. Include the following commands:

```
$ SHOW TIME
$ SHOW SYSTEM
$ SHOW ENTRY/FULL
$ SHOW QUEUE
```

1. Use one command to create and start a batch queue called username_BATCH (for example: SMITH_BATCH).

   If you get an error message indicating insufficient privilege, remind your instructor that you need OPER privilege to complete this laboratory exercise.

2. Submit B.COM on hold to be run as a batch job in your batch queue.

3. Look at the contents of the queue username_BATCH. The first line of output from a SHOW QUEUE command lists the type, name, and status of the queue. Get used to checking this line to see if the queue is stopped or paused.

   For the rest of the laboratory exercise, look at the contents of all batch queues whenever you change anything.

4. Create and start a second batch queue called SLO_username (for example: SLO_SMITH) with a base priority of 2. This time use separate commands to create and start the queue.

5. Submit B.COM as a batch job to the SLO_username queue.

6. Use the SHOW SYSTEM command to see the name of the batch process executing B.COM.

7. Submit B.COM on hold to the SLO_username queue.

8. Submit B.COM to the SLO_username queue.

9. Submit B.COM on hold to the username_BATCH queue.

10. Transfer all jobs from the username_BATCH queue to the SLO_username queue.

11. Stop the username_BATCH and SLO_username queues.

12. Delete the holding entries in the SLO_username queue.

13. Delete the username_BATCH and SLO_username queue. List all batch queues.

Solutions to Laboratory Exercise 3-2 — Batch Queues

1. Use one command to create and start a batch queue called username_BATCH.

```
$ INITIALIZE/QUEUE/BATCH/START username_BATCH
```

2. Submit B.COM on hold to be run as a batch job in the queue username_BATCH.

```
$ SUBMIT/HOLD/QUEUE=username_BATCH B.COM
```

The SUBMIT command sends jobs to the SYS$BATCH queue by default, so the following command does the same thing:

```
$ SUBMIT/HOLD B.COM
```

3. Look at the contents of the queue username_BATCH. The first line of output from a SHOW QUEUE command lists the type, name, and status of the queue. Get used to checking this line to see if the queue is stopped or paused.

```
$ SHOW QUEUE username_BATCH
```

The following commands give additional information:

```
$ SHOW QUEUE/FULL username_BATCH
```

The status column indicates the status of the job you submitted.

4. Create and start a batch queue called SLO_username with a base priority of 2. This time use separate commands to create and start the queue.

```
$ INITIALIZE/QUEUE/BATCH/BASE_PRIORITY=2 SLO_username
$ START/QUEUE SLO_username
$ SHOW QUEUE/FULL/BATCH
```

5. Submit B.COM as a batch job to the SLO_username queue.

```
$ SUBMIT/QUEUE=SLO_username B.COM
$ SHOW QUEUE/FULL SLO_username
```

Notice that the priority of the job is different from the priority of the queue. Again, note the status of the job you submitted.

6. Use the SHOW SYSTEM command to see the name of the batch process executing B.COM.

```
$ SHOW SYSTEM
```

The process name is BATCHxxx, where xxx is the job entry number.

7. Submit B.COM on hold to the SLO_username queue.

```
$ SUBMIT/HOLD/QUEUE=SLO_username B.COM
$ SHOW QUEUE/FULL/BATCH
```

The order of the qualifiers is not important, as long as they are all included in the command.

8. Submit B.COM to the SLO_username queue.

```
$ SUBMIT/QUEUE=SLO_username B.COM
$ SHOW QUEUE/FULL SLO_username
```

9. Submit B.COM on hold to the username_BATCH queue.

```
$ SUBMIT/HOLD/QUEUE=username_BATCH B.COM
$ SHOW QUEUE/FULL/BATCH
```

10. Transfer all jobs from the username_BATCH queue to the SLO_username queue.

```
$ ASSIGN/MERGE SLO_username username_BATCH
$ SHOW QUEUE/FULL/BATCH
```

Notice that all jobs have been transferred to the SLO_username queue.

11. Stop the username_BATCH and SLO_username queues.

```
$ STOP/QUEUE/NEXT SLO_username
$ STOP/QUEUE/NEXT username_BATCH
$ SHOW QUEUE/FULL/BATCH
```

12. Delete the holding entries in the SLO_username queue.

```
$ DELETE/ENTRY=entry-number SLO_username
```

Repeat this command for each entry on hold in the queue.

13. Delete the username_BATCH and SLO_username queue. List all batch queues.

```
$ DELETE/QUEUE SLO_username
$ DELETE/QUEUE username_BATCH
$ SHOW QUEUE/FULL/BATCH
```

Laboratory Exercise 3-3 — Print Queues

The following exercises allow you to practice DCL commands needed to create, manipulate, view, and delete print queues. Keep in mind that the commands used are similar to the commands for batch queues, except they are missing the /BATCH qualifier.

Preparation for this lab exercise:

Copy your B.COM file to A.DAT so you will have two files to work with.

1. Create and start a generic queue named SYS$PRINT.
2. Create a print execution queue. SYS$PRINT sends jobs to the queue you create. The print execution queue should be given the name of the fastest printer on your system (or the printer assigned to you for the course). All jobs printed through this queue should include a flag page. Start the queue.
3. Create the symbol SQ to look at the contents of each device queue.

   ```
   $ SQ=="SHOW QUEUE/FULL/DEVICE"
   ```

   Use this symbol instead of the command. The symbol is faster to enter, and you will learn more about queues if you look at their contents often (especially after each change).
4. Send A.DAT on hold to the SYS$PRINT generic queue to be printed.
5. Send B.COM directly to the print execution queue to be printed. Look at the contents of each device queue (quickly).
6. Release the A.DAT job in the SYS$PRINT queue so that it will be printed.
7. Look at the contents of the device queues, and check the line printer output.
8. While the files are printing, pretend that the line printer is jammed. Fix the line printer, and requeue the file.
9. Abort the current job printing on the line printer. Display the contents of the queue and note the status of the job you just aborted.
10. Enable your terminal as an operator's console.
11. Print the file A.DAT on SYS$PRINT, using the /OPERATOR qualifier to send a message to the operator.
12. When notified operator assistance is requested, issue a SHOW QUEUE/FULL command on the print execution queue. Note the first line, which reports status of the device.
13. Start the queue again.
14. Display the contents of the execution queue.

Solutions to Laboratory Exercise 3-3 — Print Queues

The *n* in some of the answers corresponds to the entry number given to the job when the PRINT command was typed. Each job has a unique entry number. When you type these commands to see the answers, substitute the entry numbers given by your system in the appropriate places.

The device name LPA0: can be replaced by the device name of the printer you are using so the command will work on your system.

1. Create and start a generic queue named SYS$PRINT.

   ```
   $ INITIALIZE/QUEUE/GENERIC/START SYS$PRINT
   ```

2. Create a print execution queue. SYS$PRINT sends jobs to the queue you create. The print execution queue should be given the name of the fastest printer on your system (or the printer assigned to you for the course). All jobs printed through this queue should include a flag page. Start the queue.

   ```
   $ INITIALIZE/QUEUE/SEPARATE=FLAG LPA0
   ```

3. Create the symbol SQ to look at the contents of each device queue.

   ```
   $ SQ=="SHOW QUEUE/FULL/DEVICE"
   $ SQ
   ```

   The symbol lists the complete contents of all device queues. You should see two queues, named LPA0 and SYS$PRINT.

4. Send A.DAT on hold to the SYS$PRINT generic queue to be printed.

   ```
   $ PRINT/QUEUE=SYS$PRINT/HOLD A.DAT
   ```

   The PRINT command automatically sends jobs to the SYS$PRINT queue if it exists, so the following command does the same thing:

   ```
   $ PRINT A.DAT
   ```

5. Send B.COM directly to the print execution queue to be printed. Look at the contents of each device queue (quickly).

   ```
   $ PRINT/QUEUE=LPA0 B.COM
   $ SQ
   ```

   The contents of the queues should be A.DAT in SYS$PRINT on hold, and B.COM in LPA0 with printing status.

6. Release the A.DAT job in the SYS$PRINT queue so that it will be printed.

   ```
   $ SET ENTRY/RELEASE n
   ```

   (Where *n* is the entry number)

7. Look at the contents of the device queues, and check the line printer output.

```
$ SQ
```

A.DAT should have printing status on the LPA0 queue. B.COM should have been printed first, followed by the file A.DAT.

8. While the files are printing, pretend that the line printer is jammed. Fix the line printer, and requeue the file.

```
$ STOP/QUEUE/REQUEUE LPA0
```

9. Abort the current job printing on the line printer. Display the contents of the queue and note the status of the job you just aborted.

```
$ STOP/QUEUE/ABORT LPA0
```

10. Enable your terminal as an operator's console.

```
$ REPLY/ENABLE
```

11. Print the file A.DAT on SYS$PRINT:, using the /OPERATOR qualifier to send a message to the operator.

```
$ PRINT/OPERATOR="Long job coming through -- check paper."
```

12. When you are notified that operator assistance is requested, issue a SHOW QUEUE/FULL command on the print execution queue. Notice the first line, which reports the status of the device.

```
$ SHOW QUEUE/FULL LPA0
```

13. Start the queue again.

```
$ START/QUEUE LPA0
```

14. Display the contents of the execution queue.

```
$ SHOW QUEUE/FULL LPA0
```

LABORATORY EXERCISES — PERFORMING BACKUPS AND RESTORES

Laboratory Exercise 4-1 — Backup

This exercise allows the user to perform backup procedures without a tape drive. Perform the following exercise at an interactive terminal. Refer to the Appendix in the **Performing Backups and Restores** chapter if you need help with this exercise.

1. Create a new subdirectory one level below your main directory. Name the new subdirectory ANDY.
2. Using the BACKUP utility, copy the file A.DAT to the new subdirectory. Create a new file and **NOT** a save set.
3. Using the BACKUP utility, copy the same file A.DAT to the new subdirectory. Create a save set named BEGIN.BCK and **NOT** a file copy.
4. Using the **/LIST** qualifier, examine the contents of your newly created save set.
5. Delete both A.DAT and BEGIN.BCK from the subdirectory.
6. Create a save set named COMS.BCK in the subdirectory ANDY that contains all command procedure files in the directory SYS$EXAMPLES:. The BACKUP utility will not copy the files that you have no privilege to access.
7. Using the /LIST qualifier, verify that the backup save set contains the intended command procedure files.
8. Select several of the command procedures and copy them from the save set to the subdirectory ANDY one at a time.

Solutions to Laboratory Exercise 4-1 — Backup

Substitute your directory name for STUDENT in the following examples.

1. Create a new subdirectory one level below your main directory. Name the new subdirectory ANDY.

```
$ CREATE/DIRECTORY    [STUDENT.ANDY]
```

2. Using the BACKUP utility, copy the file A.DAT to the new subdirectory. Create a new file and **NOT** a save set.

```
$ BACKUP  A.DAT
_To:    [STUDENT.ANDY]
```

3. Using the BACKUP utility, copy the same file A.DAT to the new subdirectory. Create a save set named BEGIN.BCK and **NOT** a file copy.

```
$ BACKUP A.DAT
_To: [STUDENT.ANDY]BEGIN.BCK/SAVE_SET
```

4. Using the **/LIST** qualifier, examine the contents of your newly created save set.

```
$ BACKUP/LIST [STUDENT.ANDY]BEGIN.BCK/SAVE_SET
```

5. Delete both A.DAT and BEGIN.BCK from the subdirectory.

```
$ DELETE [STUDENT.ANDY]*.*;*
```

6. Create a save set named COMS.BCK in the subdirectory ANDY that contains all command procedure files in the directory SYS$EXAMPLES:. The BACKUP utility will not copy the files that you have no privilege to access.

```
$ BACKUP SYS$EXAMPLES:*.COM
_To:    [STUDENT.ANDY]COMS.BCK/SAVE_SET
```

7. Using the /LIST qualifier, verify that the backup save set contains the intended command procedure files.

```
$ BACKUP/LIST    [STUDENT.ANDY]COMS.BCK/SAVE_SET
```

8. Select several of the command procedures and copy them from the save set to the subdirectory ANDY one at a time.

```
$ BACKUP
_From: [STUDENT.ANDY]COMS.BCK/SAVE_SET/SELECT=file
_To:    [STUDENT.ANDY]
```

Laboratory Exercise 4-2 — Backup to Tape (Optional)

If the following equipment is available to you, perform the following procedures. Use the scratch tape provided by the instructor for the following problems.

1. Allocate a tape drive and load a tape on the drive.
2. Initialize and mount a new magnetic tape. If the tape is not new and does not belong to you, you will need VOLPRO privilege. Mount the tape using the following command:

   ```
   $ MOUNT/OVERRIDE=ID tape label
   ```

3. Copy several files to the tape.
4. List the files on the tape.
5. Dismount the tape, allowing it to rewind completely and go off line.
6. Remove the tape from the drive.
7. Mount a small tape as the first tape in a volume set.
8. Copy the files in SYS$SYSTEM: to the tape.
9. When the tape is full, the system requests another tape. Mount the next tape and allow the procedure to continue.
10. Dismount and deallocate the tape drive.

    Do the following exercises only if more than one tape drive is attached to your system.

11. Allocate two tape drives and load a tape on each.
12. Initialize the first tape in the set.
13. Mount the tapes as a multivolume set, using the /INITIALIZE=CONTINUOUS qualifier to initialize the tapes before you write to them.
14. Copy SYS$SYSTEM: to the tape set.

Solutions to Laboratory Exercise 4-2 — Backup to Tape (Optional)

1. Allocate a tape drive and load a tape on the drive.

   ```
   $ ALLOCATE MTA0:
   ```

2. Initialize and mount a new magnetic tape.

   ```
   $ INITIALIZE MTA0: label
   ```

   Mount the tape using the following command:

   ```
   $ MOUNT/OVERRIDE=ID tape label
   ```

3. Copy several files to the tape.

   ```
   $ COPY files MTA0:
   ```

4. List the files on the tape.

   ```
   $ DIRECTORY MTA0:
   ```

5. Dismount the tape, allowing it to rewind completely and go off line.

   ```
   $ DISMOUNT MTA0:
   ```

6. Remove the tape from the drive.

   See the **Handling Peripherals** chapter.

7. Mount a small tape as the first tape in a volume set.

   ```
   $ MOUNT MTA0: label
   ```

8. Copy the files in SYS$SYSTEM: to the tape.

   ```
   $ COPY SYS$SYSTEM:*.*;* MTA0:
   ```

9. When the tape is full, the system requests another tape. Mount the next tape and allow the procedure to continue.

   See chapter text.

10. Dismount and deallocate the tape drive.

```
$ DISMOUNT MTA0:
$ DEALLOCATE MTA0:
```

11. Allocate two tape drives and load a tape on each.

```
$ ALLOCATE MTA0:
$ ALLOCATE MTA1:
```

12. Initialize the first tape in the set.

```
$ INITIALIZE MTA0: label
```

13. Mount the tapes as a multivolume set, using the /INITIALIZE=CONTINUOUS qualifier to initialize the tapes before you write to them.

```
$ MOUNT/OVERRIDE=ID/INITIALIZE=CONTINUOUS MTA0:,MTA1:
```

14. Copy SYS$SYSTEM: to the tape set.

```
$ COPY SYS$SYSTEM: MTA0:
```

LABORATORY EXERCISES — INTRODUCTION TO SYSTEM CUSTOMIZATION

Laboratory Exercise 5-1 — Customization

1. Type (or print) the following system startup files and examine them:

   a. SYS$MANAGER:SYLOGICALS.COM

   b. SYS$MANAGER:SYPAGSWPFILES.COM

   c. SYS$MANAGER:SYCONFIG.COM

   d. SYS$MANAGER:SYSTARTUP_V5.COM

2. Optional - Create TERMINALS.COM in your own directory. This procedure should set the permanent characteristics of the terminals on your system according to the following information.

   a. Assume that you have eight terminals.

   b. Set up some fast terminals and some slow terminals.

   c. Protect at least three terminals from allocation by processes other than those with a [001,004] UIC.

   d. Four of the terminals are VT200 series terminals.

   e. Three are VT300 series terminals.

   f. One is an LA120 hardcopy terminal.

   g. The LA120 is attached through a modem.

   h. The other terminals are attached through direct lines.

   i. Include the name of the owner and office number where the terminal is located in a comment for each terminal.

3. Optional - Create an alternate SYSTARTUP_V5.COM in your own directory. Do not execute this procedure to verify your work. Instead, look at the answers provided in this chapter. The procedure should include commands to do the following:

   a. Mount the class disk (label = CLASS) on the device DUA1:. Be sure it will be accessible to all users. Assign the disk the logical name CLASS_DISK.

   b. Assume that a directory named PROGRAMS.DIR has been created on the system disk to contain site-specific programs. Define a system logical name for this directory.

   c. Create and start the following queues (if possible).

      - SYS$PRINT (generic queue)
      - A print execution queue
        - Make sure the printer supports lowercase characters and includes a flag page on each job printed.
      - SYS$BATCH (batch execution queue)
        - Set the job limit at two and the priority at three.
      - BIGJOB (batch execution queue)
        - Set the job limit at one and the priority at two.
        - Do not start this queue.

   d. Invoke SYS$MANAGER:TERMINALS.COM

   e. Define the logical names SYS$ANNOUNCE and SYS$WELCOME

   f. Rename the second highest version of the operator log to OPERATOR.OLD. Print OPERATOR.OLD, and have the system delete the file for you after it has been printed.

   g. Restrict the number of interactive users to 35.

   h. Send a message to all terminals telling users that the system is now up and ready for use, and log out.

Solutions to Laboratory Exercise 5-1 — Customization

1. The purpose of looking at the system startup files is to familiarize yourself with the contents of these files as a whole. The following are the commands that you would use to type out the system startup files.

   a. `$ TYPE SYS$MANAGER:SYLOGICALS.COM`

   b. `$ TYPE SYS$MANAGER:SYPAGSWPFILES.COM`

   c. `$ TYPE SYS$MANAGER:SYCONFIG.COM`

   d. `$ TYPE SYS$MANAGER:SYSTARTUP_V5.COM`

2. The following is a sample TERMINALS.COM file. This procedure meets the specifications of the problem, but it is not the only answer. If your procedure contains most of the same kinds of statements, it is probably also correct.

```
$!TERMINALS.COM
$!
$! This file sets up the permanent characteristics of the
$! terminals on this system.  This procedure is typically
$! invoked from SYSTARTUP_V5.COM.
$!
$!-------------------------------------------------------
$!
$ SET NOON
$!
$ SET TERM/PERM/NOMODEM-
/VT200/SPEED=300  TTA0:     !J.Smith E15
$ SET TERM/PERM/NOMODEM-
/VT200/SPEED=2400 TTA1:     !N.Hae E16
$ SET TERM/PERM/NOMODEM-
/VT200/SPEED=9600 TTA2:     !F.Chi E17
$ SET TERM/PERM/NOMODEM-
/VT200/SPEED=9600 TTA3:     !P.Jones E18
$ SET TERM/PERM/NOMODEM-
/VT300/SPEED=2400  TTA4:     !A.Steel E19
$ SET TERM/PERM/NOMODEM-
/VT300/SPEED=9600  TTA5:     !J.Howland F01
$ SET TERM/PERM/NOMODEM-
/VT300/SPEED=9600  TTA6:     !M.Carter F02
$ SET TERM/PERM/MODEM-
/AUTOBAUD/LA120    TTA7:     !D.Trevor Dial-up
$!
$! NOTE: For the LA120 terminal, the speed is
$!  set automatically by /AUTOBAUD to 9600.
$!
$!Give [001,004] ownership of three terminals:
$!
$ SET PROT=(S,O:R,G,W)/DEVICE/OWNER_UIC=[001,004] TTA0:
$ SET PROT=(S,O:R,G,W)/DEVICE/OWNER_UIC=[001,004] TTA5:
$ SET PROT=(S,O:R,G,W)/DEVICE/OWNER_UIC=[001,004] TTA7:
$!
$! NOTE: All users can log in on these terminals, but users
$!       with a UIC of [001,004] can also allocate them.
```

3. The following is a sample SYSTARTUP_V5.COM procedure. This procedure meets the specifications of the problem, but it is not the only answer. If your procedure contains most of the same kinds of statements, it is probably also correct.

```
$! SAMPLE_SYSTARTUP_V5.COM
$!
$!  This procedure sets up the system environment according
$!  to the resources available and functions performed on
$!  this system.
$!
$!------------------------------------------------------------
$!
$ MOUNT/SYSTEM DUA1: CLASS CLASS_DISK
$!
$ ASSIGN/SYSTEM SYS$SYSDEVICE:PROGRAMS.DIR PROGRAMS
$!
$!  Start up PRINT and BATCH queues
$!
$ INITIALIZE/QUEUE/GENERIC/START SYS$PRINT
$!
$ SET PRINTER LPA0:/LOWER
$ SET DEVICE LPA0:/SPOOLED
$!
$ INITIALIZE/QUEUE/FLAG/START LPA0
$!
$ INITIALIZE/QUEUE/BATCH/JOB=2/PRIORITY=3/START SYS$BATCH
$!
$ INITIALIZE/QUEUE/BATCH/JOB=1/PRIORITY=2 BIGJOB
$!
$ @SYS$MANAGER:TERMINALS.COM
$!
$ ASSIGN/SYSTEM "System number 239 - VMS " SYS$ANNOUNCE
$ ASSIGN/SYSTEM "Welcome to system 239" SYS$WELCOME
$!
$ RENAME OPERATOR.LOG;-1 OPERATOR.OLD
$ PRINT/DELETE OPERATOR.OLD
$!
$ SET LOGIN/INTERACTIVE=35
$!
$! NOTE: The SET LOGIN/INT command should be placed at the end
$!       of the procedure, so users cannot log in while the
$!       procedure is executing.
$!
$!
$ REPLY/BELL/ALL "System 239 is ready for use"
$!
$ LOGOUT/BRIEF
```

Laboratory Exercise 5-2 — Customization

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

1. Create an account called PAYROLLn, where n is a number assigned by your instructor. 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. Your instructor can tell you what directory these files are in.

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

3. Log in as PAYROLLn. 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 PAYROLLn 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; and so forth.)

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

Solutions to Laboratory Exercise 5-2 — Customization

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 PAYROLL3 /PASSWORD=JOE-
_UAF>/UIC=[322,063]-
_UAF>/DEVICE=student_disk-
_UAF>/DIRECTORY=[PAYROLL3]
UAF>EXIT
$
$ SET DEFAULT student_disk:[000000]
$
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA ADD PAYROLL3 /DEVICE=student_disk -
_SYSMAN> /PERMQUOTA=5000/OVERDRAFT=500
SYSMAN> EXIT
$
$ CREATE/DIRECTORY/OWNER=PAYROLL3 [PAYROLL3]
$
```

   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 student_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 PAYROLL3 /LGICMD=student_disk:[PAYROLL]DATA.COM-
_UAF>/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 [PAYROLL3], remove PAYROLL3.DIR, remove the PAYROLL3 entry from the quota file, and remove the UAF record from the SYSUAF.DAT file.

Laboratory Exercise 5-3 — Customization

In these exercises, you use a command procedure called HARMLESS_AUTOGEN.COM. Your instructor should tell you in what directory to find this procedure.

HARMLESS_AUTOGEN mimics the actions of SYS$UPDATE:AUTOGEN.COM, except that it does not create or modify any system files or shut down the system. Instead, it creates files in your default login directory (SYS$LOGIN:).

1. Create a file MODPARAMS.DAT in your default directory SYS$LOGIN: (not in SYS$MANAGER:). Have it contain values for the following:
   - Preserve the values of parameters SCSNODE and SCSSYSTEMID for this system (use the values that appear in SYS$SYSTEM:MODPARAMS.DAT).
   - Set VIRTUALPAGECNT to allow a program to use 20 megabytes (40960 pages) of virtual memory.
   - Set RJOBLIM to allow only four remote terminals to log in concurrently.
   - Increase the number of global sections by six, and the number of global pages by 300.
2. Run HARMLESS_AUTOGEN so that it starts at the earliest possible phase and ends with the phase that reports on system file sizes.
3. Examine SYS$LOGIN:AGEN$FEEDBACK.REPORT and SYS$LOGIN:SETPARAMS.DAT. (You may want to print them.) Verify that AUTOGEN generated correct settings for the parameters that you specified in MODPARAMS.DAT. If settings are incorrect, edit MODPARAMS.DAT and perform the previous step again.
4. Run HARMLESS_AUTOGEN so that it starts by generating system parameter values and ends by rebooting the system. (It will not cause the system to reboot, but will display a message at the point where AUTOGEN would have rebooted the system.)
5. Use the SYSMAN utility to examine SYS$LOGIN:AUTOGEN.PAR. For the parameters that you specified in MODPARAMS.DAT, verify that AUTOGEN set them correctly in AUTOGEN.PAR.

Solutions to Laboratory Exercise 5-3 — Customization

1. Your file should be similar to the following:

```
! This is a sample MODPARAMS.DAT
!
SCSNODE = "PANAMA"
SCSSYSTEMID = 2278
MIN_VIRTUALPAGECNT = 40960 ! Use MIN in case AUTOGEN calculates a higher value
RJOBLIM = 4
ADD_GBLSECTIONS = 6
ADD_GBLPAGES = 300
```

2. `$ @dir:HARMLESS_AUTOGEN SAVPARAMS TESTFILES`

3. No solution needed.

4. `$ @dir:HARMLESS_AUTOGEN GENPARAMS REBOOT`

5.
```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> PARAMETERS USE AUTOGEN.PAR
SYSMAN> PARAMETERS SHOW SCSNODE
SYSMAN> PARAMETERS SHOW SCSSYSTEMID
SYSMAN> PARAMETERS SHOW LRPSIZE
      .
      .
      .
```

LABORATORY EXERCISES — LAYERED PRODUCT INSTALLATION

Laboratory Exercise 6-1 — Layered Product Installation

1. Type the LMF command to display a list of licenses registered in the license database for:
   a. All licenses registered
   b. A specific layered product
2. Type the LMF command that displays active licenses on the current node for:
   a. All active licenses
   b. A specific layered product

Solutions to Laboratory Exercise 6-1 — Layered Product Installation

1. The purpose of displaying the list of all registered licenses is to begin to familiarize yourself with the contents of the license database. This command allows you to gain information on the entire database or just on specific products.

   a. `$ LICENSE LIST`

   b.

```
$ LICENSE LIST FORTRAN

Press Ctrl/Z to exit, use arrow keys to scroll.
License Management Facility

License Database File:       SYS$COMMON:[SYSEXE]LMF$LICENSE.LDB;1
Created on:                   9-MAY-1989
Created by user:             SYSTEM
LMF Version:                 V1.0

----------------------------------
FORTRAN                      DEC
FORTRAN                      DEC
[End Of List]
```

2. The SHOW LICENSE command gives you the ability to find out what products have already been loaded on the current node. This command will give you the entire list of active products or you can request information on a specific product.

   a. `$ SHOW LICENSE`

   b.

```
$ SHOW LICENSE FORTRAN

Active licenses on node SUPER:

FORTRAN
        Producer: DEC
        Units: 0
        Version: 0.0
        Date: 22-AUG-1990
        Termination Date: (none)
        Availability: F (Layered Products)
        Activity: 0
        MOD_UNITS
```

Laboratory Exercise 6-2 — Layered Product Installation

1. Create a license database in your own student directory. Issue the SHOW DEFAULT command to find out the exact device and directory name that you will need to specify when creating the database.

2. Using the information from the following PAK, register and list the sample product ED_ SERV_FORTRAN. You will need to specify the /DATABASE qualifier to the REGISTER command to ensure you register it to your private database.

```
 - - - - - - -                                                  _____________________
| | | | | | | |            LICENSE SOFTWARE PRODUCT            | DOCUMENT ISSUE DATE |
|d|i|g|i|t|a|l|            PRODUCT AUTHORIZATION KEY           |     03-JUNE-1991    |
| | | | | | | |                                                 ---------------------
 - - - - - - -
Digital Equipment Corporation
Maynard, Ma.
 -------------------------------------
| LICENSE ADMINISTRATION LOCATION:    |  ORDERED BY: Moppet Industries
|                                     |              Ms. Bonnie Moppet
| Digital Equipment Corporation       |              123 Main St.
| Maynard, Massachusetts              |              Merrimack, NH 03054
|                                     |
|                                     |
|                                     |
 -------------------------------------
********************************************************************************
!                     FOR DIGITAL INTERNAL USE ONLY
PAK ID:
                   Issuer: DEC
     Authorization Number: SQM003058

PRODUCT ID:
             Product Name: ED_SERV_FORTRAN
                 Producer: DEC

NUMBER OF UNITS:
          Number of units: 500

KEY LEVEL:
                  Version: 1.0
     Product Release Date:

KEY TERMINATION DATE:
     Key Termination Date:

RATING:
  Availability Table Code:
      Activity Table Code:

MISCELLANEOUS:
              Key Options:
            Product Token:
              Hardware-Id:
                 Checksum: 1-PAOI-MJNP-NCJL-DEKP
********************************************************************************
```

Solutions to Laboratory Exercise 6-2 — Layered Product Installation

1. The following is the command that you would use to create a new license database called PRACTICE.LDB in the directory [MORGAN.STUDENT]. Your database location would be different than the one used in this solution. A DIRECTORY command issued afterwards verifies that the file was created in the requested directory.

```
$
$ LICENSE CREATE /DATABASE=$1$DUA1:[MORGAN.STUDENT]PRACTICE.LDB
$
$ DIRECTORY

Directory $1$DUA1:[MORGAN.STUDENT]

PRACTICE.LDB;1

Total of 1 file.
```

2. The following are the License commands that you would enter to register and then list the layered product ED_SERV_FORTRAN.

```
$ license register "ED_SERV_FORTRAN" -
_$ /ISSUER="DEC" -
_$ /AUTHORIZATION=SQM003058 -
_$ /PRODUCER=DEC -
_$ /UNITS=500 -
_$ /VERSION=1.0 -
_$ /CHECKSUM=1-PAOI-MJNP-NCJL-DEKP -
_$ /DATABASE=$1$DUA1:[MORGAN.STUDENT]PRACTICE.LDB
$
$ LICENSE LIST ED_SERV_FORTRAN /DATABASE=$1$DUA1:[MORGAN.STUDENT]PRACTICE.LDB
$
```

   Note that the use of the SHOW LICENSE command here will default to the actual system database, not your practice database. There is no /DATABASE qualifier to the SHOW LICENSE command.

LABORATORY EXERCISES — REPORTING ON USER ACTIVITY

Laboratory Exercise 7-1 — ACCOUNTING Utility

1. To observe how the ACCOUNTING utility works, perform the following:

   a. Find out what classes of accounting have been enabled.

   b. Disable the recording of accounting data on print jobs.

   c. Send a file to the printer.

   d. Enable the recording of accounting on print jobs.

   e. Send a file to the printer.

   f. Generate an ACCOUNTING report in brief format that shows all print jobs completed within the last hour. Observe the contents of the report.

2. Create an accounting report in full format that shows all print jobs completed within the last hour. Send the report directly to the printer. Collect the output and observe the format and contents of the report.

3. Use the ACCOUNTING utility to examine the following record types:

   a. Login failures (in brief format)

   b. Interactive job terminations (in brief format)

   c. Process terminations (in brief format)

   d. System initializations (in full format)

4. Use the ACCOUNTING utility to display a summary report of the accounting records that have your user name.

5. Most direct I/O is disk I/O, so you can use ACCOUNTING to show which users make heaviest use of the disks. Create a report in summary format containing the data collected on direct I/O from interactive processes today. List the data in order by UIC, and send the report directly to the printer. (This can all be done in the same command.)

6. Create a summary report that shows how many batch jobs have completed in each hour.

7. Find out whether anyone has tried and failed to log in to the SYSTEM account.

Solutions to Laboratory Exercise 7-1 — ACCOUNTING Utility

1. Enter the following commands to observe how the ACCOUNTING utility works:

   a. `$ SHOW ACCOUNTING`

   b. `$ SET ACCOUNTING/DISABLE=(PRINT)`

   c. `$ PRINT FILE.TXT`

      Substitute the name of your file for FILE.TXT.

   d. `$ SET ACCOUNTING/ENABLE=(PRINT)`

   e. `$ PRINT FILE.TXT`

   f. `$ ACCOUNTING/TYPE=PRINT/SINCE=14:00`

      Substitute an appropriate time for 14:00. Note that the report contains a brief description of your print jobs (if they have already completed) as well as descriptions of other jobs.

2. `$ ACCOUNTING/FULL/TYPE=PRINT/SINCE=14:00/OUTPUT=LPA0:`

   Substitute an appropriate time for 14:00 and the device name of your line printer for LPA0:. The colon (:) indicates that LPA0: is a device and not a file specification. If the colon is omitted, the output from the report is stored in a file named LPA0.LIS.

3. Enter the following commands to use the ACCOUNTING utility:

   a. `$ ACCOUNTING/TYPE=LOGFAIL`

   b. `$ ACCOUNTING/PROCESS=INTERACTIVE`

   c. `$ ACCOUNTING/TYPE=PROCESS`

   d. `$ ACCOUNTING/FULL/TYPE=SYSINIT`

4. `$ ACCOUNTING /SUMMARY=USER /REPORT=RECORDS /USER=SMITH`

   Substitute your user name for SMITH.

5. ```
   $ ACCOUNTING /SUMMARY=UIC -
   _$ /REPORT=DIRECT_IO /SINCE=00:00 /OUTPUT=LPA0:
   ```

   Substitute the device name of your line printer for LPA0: The information in the report is organized in order by UIC.

6. `$ ACCOUNTING /TYPE=BATCH /SUMMARY=HOUR`

7. `$ ACCOUNTING /USER=SYSTEM /TYPE=LOGFAIL`
   ```

LABORATORY EXERCISES — MAINTAINING SYSTEM SECURITY

Laboratory Exercise 8-1 — Passwords

NOTE

You must have write access to the system authorization file and SECURITY privilege to do this lab.

1. Create a secondary password for your UAF record. Log out and log in. What has changed?
2. Coordinate with the other students the creation of a system password and set the appropriate characteristics of your terminal so it is a system-password-required terminal. Log out and log in. What has changed?
3. Create another UAF record. Set the expiration date on the new UAF record to be five minutes after the current time. Wait at least five minutes. Attempt to log in to the new account. What happens? Log in to your own account and delete the extra UAF record.
4. Change the minimum length of the password to 16 on your own account, set a flag to require password generation, exit from the utility, log out and log in.
5. Change your password with a DCL command that uses one of the choices listed by the generator (or generate a new list).
6. Change your password and then try to change it again using the same password.
7. Try to change your password to any English word.
8. Log out and attempt to log in several times, specifying an incorrect password each time. Observe the output on the console terminal.
9. Using SYSMAN, adjust the SYSGEN parameter LGI_RETRY_LIM and observe the effect.

Solutions to Laboratory Exercise 8-1 — Passwords

1. ```
   $ SET DEFAULT SYS$SYSTEM
   $ RUN AUTHORIZE
   UAF> MODIFY JONES/PASSWORD=(" ",SECRET)
   ```

   The system displays a second password prompt when you log in.

2. Use the SET PASSWORD/SYSTEM command to create a system password. Use the SET TERMINAL command to modify your terminal characteristics as shown in the chapter. When you log in to your account after setting these up, you must enter the system password before you see the prompt for your user name.

3. You will not be able to log in to the new account because it has expired.

4. Use the /FLAGS=GENPWD qualifier to the MODIFY command in the AUTHORIZE utility to do this.

5. When you enter the SET PASSWORD command, it automatically generates a list of passwords for you to choose from.

6. When you set your password it adds that password to your password history file. The next time you set your password it checks the password you type against the password history file; if it finds the password there it displays the error message.

   ```
   $ SET PASSWORD
   Old password:
   New password:
   Verification:
   %SET-F-PWDNOTDIF, new password must be different from current password
   ```

7. When you type a new password, the first thing the system does (even before it prompts you to type the new password verification) is to check the system password dictionary. If it finds the password you typed in the dictionary, you will receive an error message.

   ```
   $ SET PASSWORD
   Old password:
   New password:
   %SYSTEM-F-PWDINDIC, password found in system dictionary; please choose another
   string
   ```

8. No answer needed.
   ```

9. When you modify the LGI_RETRY_LIM, the system restricts the user to that number of retries before making the user start the login process over again.

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> PARAMETER USE ACTIVE
SYSMAN> PARAMETERS SET LGI_RETRY_LIM 4
SYSMAN> PARAMETERS SHOW LGI_RETRY_LIM

Node BOUNTY:   Parameters in use: ACTIVE
Parameter Name            Current   Default   Minimum   Maximum Unit  Dynamic
--------------            -------   -------   -------   ------- ----  -------
LGI_RETRY_LIM                   4         3         0       255 Tries       D

SYSMAN> EXIT
```

Laboratory Exercise 8-2 — Intrusion Records

Team up with another student for this exercise. One student should use an unprivileged (UPRIVn) account and the other should use a privileged (SECn) account. In the following exercises you will try to become first a *suspect* and then an *intruder* from the UPRIVn account.

1. From the UPRIVn account, SET HOST 0 and enter an invalid password and/or user name three times or until control is returned to the host node.
2. From the SECn account, enter SHOW INTRUSION. There should be one *suspect* record, which you caused.
3. Repeat this three more times or until control is returned to the host node.
4. From the SECn account, enter SHOW INTRUSION. There should be one *intruder* record, which you caused.
5. From the UPRIVn account, try to SET HOST 0 and log in with the correct user name /password to the UPRIVn account.

   Did it work? Why or why not?
6. From the SECn account, try to SET HOST 0 and log in to the UPRIVn account with the correct user name/password.

   Did it work? Why or why not?

Solutions to Laboratory Exercise 8-2 — Intrusion Records

In the following exercises you will try to become first a *suspect* and then an *intruder* from the UPRIVn account.

1. From the UPRIVn account, SET HOST 0 and enter an invalid password and/or user name three times or until control is returned to the host node.

```
$ SET HOST 0

  Username: GOSSE
  Password:
  User authorization failure

  Username: MOSSE
  Password:
  User authorization failure

  Username: KANGO
  Password:
  User authorization failure
  %REM-S-END, control returned to node _SUPER::
```

2. From the SECn account, enter SHOW INTRUSION. There should be one *suspect* record, which you caused.

```
$ SHOW INTRUSION
  Intrusion       Type      Count  Expiration    Source
    NETWORK      SUSPECT      3   15:48:37.01  SUPER::UPRIVN
```

3. Repeat this three more times or until control is returned to the host node.

```
$ SET HOST 0

  Username: SWAN
  Password:
  User authorization failure

  Username: GORILLA
  Password:
  User authorization failure

  Username: KOALA
  Password:
  User authorization failure
  %REM-S-END, control returned to node _SUPER::
```

4. From the SECn account, enter SHOW INTRUSION. There should be one *intruder* record, which you caused.

```
$ SHOW INTRUSION
  Intrusion       Type       Count  Expiration    Source
    NETWORK      INTRUDER      7   15:40:49.57  SUPER::UPRIVN
```

5. From the UPRIVn account, try to SET HOST 0 and log in with the correct user name /password to the UPRIVn account.

   Did it work? Why or why not?

```
$ SET HOST 0

  Username: UPRIVN
  Password:                    !enter correct password
  User authorization failure
```

   It did not work because the INTRUDER record is for any REMOTE (SET HOST) access originating from your node and user UPRIVn.

6. From the SECn account, try to SET HOST 0 and log in to the UPRIVn account with the correct user name/password.

   Did it work? Why or why not?

```
$ SET HOST 0

  Username: UPRIVN
  Password:                    !enter correct password
```

   Yes. The INTRUDER record specifies the source of the REMOTE login, not the target account. So UPRIVn cannot set host to anyone, but you can SET HOST to the uprivn account.

Laboratory Exercise 8-3 — Suspect Versus Intrusion Records

1. Create some more suspect and intruder records.

   a. DIR 0"UPRIVn pwd":: — use your correct password

   b. DIR 0"UPRIVn XXX":: — repeat with a different user name/password pair six or more times

   c. DIR 0"UPRIVn pwd":: — use your correct password

      Did it work?

2. Try logging in locally to the node you are on several times with any combination of incorrect user names and passwords.

3. Enter the commands:

   a. SHOW INTRUSION/TYPE=SUSPECT

   b. SHOW INTRUSION/TYPE=INTRUDER

   c. SHOW INTRUSION

   What is the default qualifier value for /TYPE?

4. Find an INTRUDER record relating to you and delete the record. Verify it is gone by entering SHOW INTRUSION/TYPE=ALL.

   Now try to get in through that source (that is, network, remote, local, and so on).

   Did it work? Why or why not?

Solutions to Laboratory Exercise 8-3 — Suspect Versus Intrusion Records

1. Create some more suspect and intruder records.

   a. DIR 0"UPRIVn pwd":: — use your correct password

   b. DIR 0"UPRIVn XXX":: — repeat with a different user name/password pair six or more times

   c. DIR 0"UPRIVn pwd":: — use your correct password

      Did it work?

   No, it did not work. The INTRUDER record is for you, as a network source (node::UPRIVn). Therefore, any network access from that account will be denied.

2. Try logging in locally to the node you are on several times with any combination of incorrect user names and passwords.

3. Enter the commands:

   a. SHOW INTRUSION/TYPE=SUSPECT

   b. SHOW INTRUSION/TYPE=INTRUDER

   c. SHOW INTRUSION

   What is the default qualifier value for /TYPE?

   The default qualifier is /TYPE=ALL.

4. Find an INTRUDER record relating to you and delete the record. Verify it is gone by entering SHOW INTRUSION/TYPE=ALL.

```
$ SHOW INTRUSION/TYPE=INTRUDER
```

Find the record with your user name as the SOURCE.

```
$ DELETE/INTRUSION  UPRIVn
```

Now try to get in through that source (that is, network, remote, local, and so on).

Did it work? Why or why not?

Yes, it did work. You are no longer an INTRUDER. The intruder record is gone.

Laboratory Exercise 8-4 — Alternate SYSUAF (Optional)

1. Your instructor will create an alternate copy of the SYS$SYSTEM:SYSUAF.DAT file and will modify your account in the new file. The alternate copy is SYS$SYSTEM:SYSUAFALT.DAT. Start the system conversationally, specifying the alternate UAF file. Log in and enter the SHOW PROCESS command to verify that the alternate file was used. The display should contain the new values entered by your instructor. Check with your instructor to verify this.

2. Your instructor will shut the system down for you. Start it conversationally, specifying the normal SYS$SYSTEM:SYSUAF.DAT file as the UAF file. Log in and enter the SHOW PROCESS command. The display should look normal.

Solutions to Laboratory Exercise 8-4 — Alternate SYSUAF (Optional)

1. Boot the system conversationally. Enter the commands SET UAFALTERNATE 1 and CONTINUE at the SYSBOOT> prompt.

2. Boot the system conversationally. Enter the commands SET UAFALTERNATE 0 and CONTINUE at the SYSBOOT> prompt.

LABORATORY EXERCISES — MANAGING A NETWORK NODE

Laboratory Exercise 9-1 — NCP Utility

Your instructor will provide you with a node name and address to use for this exercise.

At an interactive terminal, perform the following tasks:

1. Try to use the SET HOST command to connect to the node whose name you were given. What happens?
2. Run NCP and enter commands to find out whether that node is registered in both the permanent and volatile remote node databases.
3. Define the node in the permanent database.
4. Exit from NCP and try step 1 again. What happens and why?
5. Run NCP again and enter a command to copy the node name and address from the permanent to the volatile database.
6. Exit from NCP and try step 1 again. What happens and why?
7. Run NCP and remove the node entry from both databases.

Solutions to Laboratory Exercise 9-1 — NCP Utility

These solutions illustrate the use of node name SHIRT and address 5.123. Substitute the name and address that your instructor gave you.

1. You see this message:

```
%SYSTEM-F-NOSUCHNODE, remote node is unknown
```

2. If the node is in neither database, you see this output:

```
$ RUN SYS$SYSTEM:NCP
NCP> LIST NODE SHIRT

Node Volatile Summary as of 15-SEP-1991 10:53:28

%NCP-W-UNRCMP, Unrecognized component , Node

NCP> SHOW NODE SHIRT

Node Volatile Summary as of 15-SEP-1991 10:53:28

%NCP-W-UNRCMP, Unrecognized component , Node
```

3.

```
NCP> DEFINE NODE SHIRT ADDRESS 5.123
```

or

```
NCP> DEFINE NODE 5.123 NAME SHIRT
```

4.

```
NCP> CTRL/Z
$ SET HOST SHIRT
%SYSTEM-F-NOSUCHNODE, remote node is unknown
```

The node is still not defined in the volatile database, and the volatile database is where the DECnet software gets its data during operation.

5.

```
$ RUN SYS$SYSTEM:NCP
NCP> SET NODE SHIRT ALL
```

or

```
$ RUN SYS$SYSTEM:NCP
NCP> SET NODE 5.123 ALL
```

or

```
$ RUN SYS$SYSTEM:NCP
NCP> SET NODE SHIRT ADDRESS 5.123
```

or

```
$ RUN SYS$SYSTEM:NCP
NCP> SET NODE 5.123 NAME SHIRT
```

6. If the node exists and is running, you see the login prompt. If it does not exist or is not running, you see the following message:

```
%SYSTEM-F-UNREACHABLE, remote node is not currently reachable
```

7.

```
NCP>CLEAR NODE SHIRT ALL
NCP>PURGE NODE SHIRT ALL
```

or

```
NCP>CLEAR NODE 5.123 ALL
NCP>PURGE NODE 5.123 ALL
```

Laboratory Exercise 9-2 — NCP in a VAXcluster System

1. Find out whether the executor database is in the node-specific or cluster-common directory. Where is it and why?

2. Find out whether the remote node database is in the node-specific or cluster-common directory. Where is it and why?

3. If the remote node database is in the cluster-common directory, continue with this exercise.

   Using the same node name and address you were given in Exercise 9-1, add the node to both the permanent and the volatile databases.

4. Use either the TELL or SET EXECUTOR command to set the executor node to another node in the same cluster. Find out whether the node you added appears in the permanent and volatile databases.

5. Still using that other node as the executor, copy the node entry from the permanent to the volatile database.

Solutions to Laboratory Exercise 9-2 — NCP in a VAXcluster System

1. Enter these commands:

```
$ DIRECTORY SYS$SPECIFIC:[SYSEXE]NETNODE_LOCAL.DAT
$ DIRECTORY SYS$COMMON:[SYSEXE]NETNODE_LOCAL.DAT
```

   The executor database should be in SYS$SPECIFIC:[SYSEXE]. It contains the local node's name and address, which must be unique within the network; therefore, each node must have a private copy of the executor database.

2. Enter these commands:

```
$ DIRECTORY SYS$SPECIFIC:[SYSEXE]NETNODE_LOCAL.DAT
$ DIRECTORY SYS$COMMON:[SYSEXE]NETNODE_LOCAL.DAT
```

   On many systems, the remote node database is in SYS$COMMON:[SYSEXE]. It takes less effort to maintain a single remote node database than to maintain a copy for each node in the cluster, since in most cases they all need to contain the same list of nodes.

3.

```
$ RUN SYS$SYSTEM:NCP
NCP> DEFINE NODE SHIRT ADDRESS 5.123
NCP> SET NODE SHIRT ALL
```

   or equivalent commands (see Exercise 9-1).

4. Substitute the name of the other node for JACKET in this example. You probably need to use access control to perform this operation.

```
NCP> SET EXECUTOR JACKET"username password"
NCP> LIST NODE SHIRT

Node Permanent Summary as of 15-SEP-1991 11:18:58

Remote node =   5.123 (SHIRT)

No information available

NCP>SHOW NODE SHIRT

Node Volatile Summary as of 15-SEP-1991 11:19:03

%NCP-W-UNRCMP, Unrecognized component , Node
```

   The output indicates that it is in the permanent but not the volatile database.

5.

```
NCP> SET NODE SHIRT ALL
```

   or

```
NCP> SET NODE 5.123 ALL.
```

LABORATORY EXERCISES — SYSTEM MONITORING

Laboratory Exercise 10-1 — MONITOR Utility

1. Using a video terminal, run the MONITOR utility to display the MONITOR> prompt. At the prompt, enter a command to the utility to display the PAGE class with an update interval of ten seconds. Allow the utility to run for five to ten minutes. Watch the display and notice how (if you are on an active system) the display changes with time.
2. Return to the MONITOR> prompt and display the SYSTEM class.
3. Return to the MONITOR> prompt and display the I/O class.
4. Use MONITOR to create a file called SUMMARY.DAT that contains summary information about the PROCESSES class. Allow the utility to write several screens of information to the file before you enter the CTRL/Z key sequence to return to the DCL prompt. (Each time the values in the display change, the utility writes a copy of the display to the file.) Display the file on your terminal screen after you exit from the utility.
5. Issue the command to monitor the cluster. This may take a while to build a display because a network process must be created on each node.
6. Issue the DCL command to display the cluster information once, then return to the DCL prompt. Select one other node name to be used for the next step.
7. Monitor your system plus the other selected node.
8. Have a lab partner monitor your node, then pause by returning to the MONITOR prompt. While their MONITOR display is alternately active and inactive, issue the DCL command to find out the priorities of this created process. Note the process name/priority as opposed to the PID.
9. Return to the MONITOR> prompt and display any other classes that you want. Use the HELP facility in the MONITOR utility to list the names of the classes and possible qualifiers.
10. Use the SHOW USERS command to obtain a list of the interactive users on the system. Use the SHOW SYSTEM command to obtain a list of the processes on the system. Compare the names of the interactive users with the list of process names. Note which processes appear in one but not the other.
11. Use the SHOW PROCESS/CONTINUOUS command to examine your own process. What image does the program indicate you are running? Try the V key, (get help).
12. Which DCL command gives you information on physical memory and usage?
13. Which DCL command gives you information on the page files and swap files and their usage?

Solutions to Laboratory Exercise 10-1 — MONITOR Utility

1.

```
$ MONITOR
MONITOR>  MONITOR PAGE
```

2.

```
CTRL/C
MONITOR>  MONITOR SYSTEM
```

3.

```
CTRL/C
MONITOR>  MONITOR IO
```

4.

```
CTRL/C
MONITOR>  MONITOR PROCESS/SUMMARY=SUMMARY.DAT
CTRL/Z
$ TYPE SUMMARY.DAT
```

5.

```
MONITOR>  MONITOR CLUSTER
```

6.

```
CTRL/Z
$ SHOW CLUSTER
```

7.

```
$ MONITOR CLUSTER/NODE=(yours,selected_other)
```

8.

```
CTRL/Z
                    ...as the MONITOR CLUSTER process begins

$ SHOW SYSTEM/NETWORK

VAX/VMS V5.3-2  on node WIZARD 27-JUN-1991 13:03:04.10   Uptime  26 12:12:38
  Pid    Process Name    State  Pri      I/O       CPU       Page flts Ph.Mem
20209704 FAL_25005       LEF      6     6675   0 00:00:13.64      1466     287   N
2020D090 SERVER_004A     LEF      6     4394   0 00:00:28.65     13648     304   N
2020C939 AFP_FIRLAND     LEF      6     2379   0 00:00:11.40       577     789   N
2020A0C9 FAL_8457        LEF      6     4140   0 00:00:08.86      1578     289   N
202077CE FAL_324         LEF      6     2906   0 00:00:09.17       819     332   N
2020CED4 VPM_24906       LEF      5       26   0 00:00:00.22       166     244   N
```

```
            ...after the MONITOR CLUSTER process starts collecting data

$ SHOW SYSTEM/NETWORK

VAX/VMS V5.3-2  on node WIZARD 27-JUN-1991 13:09:30.12   Uptime  26 12:19:04
  Pid    Process Name    State  Pri      I/O       CPU       Page flts Ph.Mem
20209704 FAL_25005       LEF      6     6675   0 00:00:13.64      1466    287  N
2020D090 SERVER_004A     LEF      6     4518   0 00:00:29.81     14582    300  N
2020C939 AFP_FIRLAND     LEF      6     2391   0 00:00:11.49       577    789  N
2020A0C9 FAL_8457        LEF      6     4140   0 00:00:08.87      1578    289  N
202077CE FAL_324         LEF      6     2906   0 00:00:09.17       819    332  N
2020CED4 MONITOR_SERVER  HIB     15      196   0 00:00:01.52       781    338  N

      ...after the MONITOR CLUSTER process is interrupted by a CTRL/Z

$ SHOW SYSTEM/NETWORK

VAX/VMS V5.3-2  on node WIZARD 27-JUN-1991 13:09:01.04   Uptime  26 12:18:35
  Pid    Process Name    State  Pri      I/O       CPU       Page flts Ph.Mem
20209704 FAL_25005       LEF      6     6675   0 00:00:13.64      1466    287  N
2020D090 SERVER_004A     LEF      6     4518   0 00:00:29.81     14582    300  N
2020C939 AFP_FIRLAND     LEF      6     2390   0 00:00:11.49       577    789  N
2020A0C9 FAL_8457        LEF      6     4140   0 00:00:08.87      1578    289  N
202077CE FAL_324         LEF      6     2906   0 00:00:09.17       819    332  N
2020CED4 SERVER_0006     LEF      4      181   0 00:00:01.39       643    336  N

                    ...after the MONITOR utility is terminated

$ SHOW SYSTEM/NETWORK

VAX/VMS V5.3-2  on node WIZARD 27-JUN-1991 13:20:34.38   Uptime  26 12:30:08
  Pid    Process Name    State  Pri      I/O       CPU       Page flts Ph.Mem
20209704 FAL_25005       LEF      6     6675   0 00:00:13.64      1466    287  N
2020D090 SERVER_004A     LEF      4     5004   0 00:00:33.94     17038    283  N
2020C939 AFP_FIRLAND     LEF      6     2412   0 00:00:11.50       577    789  N
2020A0C9 FAL_8457        LEF      6     4144   0 00:00:08.87      1578    289  N
202077CE FAL_324         LEF      6     2906   0 00:00:09.17       819    332  N
```

9. Enter the CTRL/C key sequence to cancel any display and return to the MONITOR> prompt.

```
$ MONITOR

MONITOR>  HELP MONITOR
```

10. There should be more processes listed by SHOW SYSTEM than by SHOW USERS. SHOW SYSTEM lists the system processes, interactive processes, processes from batch queues, and subprocesses. SHOW USERS lists interactive users attached to terminals.

11. SHOW PROCESS/CONTINUOUS shows that you are executing the image SYS$SYSROOT:[SYSEXE]SHOW.EXE.

12.

```
S SHOW MEMORY/PHYSICAL
```

13.

```
S SHOW MEMORY/FILES/FULL
```

LABORATORY EXERCISES — DEVELOPING COMMAND PROCEDURES

Laboratory Exercise 11-1 — Adding a User

Write a simple command procedure to add a user to the system.

The procedure should prompt for:

- User name
- Full name
- Password
- UIC group and member number
- Account name
- Login directory and device
- Disk quota and overdraft

Using the values provided by the user, the procedure should:

- Add a record to the UAF for the user
- Display the information in the new record
- Add a disk quota entry for the user
- Create a directory owned by the user

Hint

Some utilities, including AUTHORIZE and SYSMAN, allow you to execute a single utility command as a DCL command line. First, define a symbol that runs the utility, using this special format (called a *foreign command*):

```
$ AUTHORIZE := $AUTHORIZE
$ SYSMAN := $SYSMAN
```

Then place the utility command after the symbol name on a command line in order to execute the command, for example:

```
$ AUTHORIZE SHOW username
$ AUTHORIZE ADD username /PASSWORD=password /OWNER="owner"
$ SYSMAN DISKQUOTA SHOW username
$ SYSMAN DISKQUOTA ADD username /PERMQUOTA=blocks
```

After the command executes, control returns to DCL rather than remaining in the utility.

Solution to Laboratory Exercise 11-1 — Adding a User

```
$!******************
$!* Comment Section *
$!******************
$!   ADDUSER.COM
$!
$!   This procedure assists you in adding a user to the system. It adds
$!   the record to AUTHORIZE, adds a DISKQUOTA record, and
$!   creates a directory.
$!   For the sake of clarity, no extensive error checking is performed
$!   and no defaults are supplied for the user input.
$!
$!*************************
$!*  Initialization Section *
$!*************************
$!
$ AUTHORIZE = "$AUTHORIZE"
$ SYSMAN    = "$SYSMAN"
$!
$!***************
$!* Main Section *
$!***************
$!
$! Request Account Information
$!
$ READ SYS$COMMAND /PROMPT= "Username                       : "  USERNAME
$ READ SYS$COMMAND /PROMPT= "Full Name                      : "  FULLNAME
$ READ SYS$COMMAND /PROMPT= "Password                       : "  PASSWORD
$!
$ Get UIC
$!
$ READ SYS$COMMAND /PROMPT= "Group UIC (number or an *) : "  GROUP
$ READ SYS$COMMAND /PROMPT= "Member number                  : "  MEMBER
$ UIC = "[" + GROUP + "," + MEMBER + "]"
$ READ SYS$COMMAND /PROMPT= "Account Name                   : "  ACCOUNT
$ READ SYS$COMMAND /PROMPT= "Login Directory (use brackets) : "  DIR
$ READ SYS$COMMAND /PROMPT= "Login Device (use colon)       : "  DEVICE
$!
$ AUTHORIZE ADD 'USERNAME' /OWNER="''FULLNAME'" -
   /ACCOUNT="''ACCOUNT'" /PASSWORD='PASSWORD' /DEVICE='DEVICE'-
   /DIRECTORY='DIR' /UIC='UIC'
$!
$ WRITE SYS$OUTPUT "This is the user you added."
$ AUTHORIZE SHOW 'USERNAME'
$!
$ READ SYS$COMMAND /PROMPT= "Disk Quota                     : "  DISK_QUOTA
$ READ SYS$COMMAND /PROMPT= "Disk Overdraft                 : "  OVERDRAFT
$ SYSMAN DISKQUOTA ADD 'UIC' -
 /PERM='DISK_QUOTA'/OVERDRAFT='OVERDRAFT'/DEVICE='DEVICE'
$!
$ ADD_DIRECTORY:
$!
$ CREATE /DIRECTORY /OWNER='UIC' 'DEVICE''DIR' /LOG
$ EXIT
```

Laboratory Exercise 11-2 — Generating Accounting Reports

1. Write a short command procedure that generates an accounting report. The procedure should run at midnight every night and report on the previous day's activity.

   The report should summarize for each user:

   - Pages printed
   - Processor time
   - Peak physical memory usage (working set)
   - Direct I/O operations

   The procedure should then mail the report to the SYSTEM account and then delete the report.

2. Modify the procedure so that on Mondays it performs one additional task: it should close the current accounting file and open a new one.

   Also, if it does not already do so, it should put the day of the week in the subject heading of the mail message.

Hints

The MAIL command takes two parameters (file specification and recipient name), and has a qualifier by which you can specify the subject of a message. See the documentation or on-line help for examples of the use of these parameters and qualifier.

To determine the day of the week, your procedure needs to use the F$CVTIME lexical function. Use the documentation or on-line help to find examples of the usage of F$CVTIME.

Solutions to Laboratory Exercise 11-2 — Generating Accounting Reports

1.

```
$!ACCOUNTING_REPORT.COM
$!
$! This procedure generates a nightly accounting report and mails
$! it to the system manager's account.
$!
$ ACCOUNTING /SUMMARY=USER /REPORT=(PROCESSOR,WORKING_SET,DIRECT_IO)-
 /SINCE=YESTERDAY /OUTPUT=NIGHTLY_REPORT.LIS
$ MAIL NIGHTLY_REPORT.LIS SYSTEM /SUBJECT="Nightly accounting report"
$ DELETE NIGHTLY_REPORT.LIS
$ SUBMIT /AFTER=TOMORROW SYS$MANAGER:ACCOUNTING_REPORT.COM
```

2.

```
$!ACCOUNTING_REPORT.COM
$!
$! This procedure generates a nightly accounting report and mails
$! it to the system manager's account.
$!
$! It also closes the accounting file every Monday and opens a new one.
$!
$ DAY = F$CVTIME(,,"WEEKDAY")
$ ACCOUNTING /SUMMARY=USER /REPORT=(PROCESSOR,WORKING_SET,DIRECT_IO)-
 /SINCE=YESTERDAY /OUTPUT=NIGHTLY_REPORT.LIS
$ IF DAY .EQS. "Monday" THEN SET ACCOUNTING /NEW_FILE
$ MAIL NIGHTLY_REPORT.LIS SYSTEM /SUBJECT="Nightly accounting report for ''DAY'"
$ DELETE NIGHTLY_REPORT.LIS
$ SUBMIT /AFTER=TOMORROW SYS$MANAGER:ACCOUNTING_REPORT.COM
```

Laboratory Exercise 11-3 — Writing a Backup Procedure

Write a command procedure that resubmits itself to run every night at midnight. It backs up disk 1DUA1 to tape drive 1MUA0.

It should perform an incremental backup unless it is Wednesday, in which case it should perform a full backup.

Define the disk and tape device names as symbols near the beginning of your procedure, so that you could easily change them.

Solution to Laboratory Exercise 11-3 — Writing a Backup Procedure

```
$!Comment Section
$!
$!    BACKUP.COM
$!                This procedure assists the operator in
$!                performing incremental and full backups
$!    ASSUMPTION:
$!                The assumption is that FULL backups are done
$!                on Wednesday. The procedure tests for the day
$!                of the week and acts accordingly.
$!
$!Initialization Section
$!
$!
$ DISK = "$1$DUA1:"
$ TAPE = "MUA0:"
$!
$!Main Section
$!
$ DAY = F$CVTIME(,,"WEEKDAY")
$ IF DAY .EQS. "Wednesday" THEN
$!
$!  Do full backup
$!
$  ALLOCATE MUA0:
$  INITIALIZE MUA0: FULLBK
$  BACKUP /REWIND /RECORD /DENSITY=1600 /BLOCK=32784 -
  /IMAGE 'DISK' MUA0:FULLBK.BCK
$  WRITE SYS$OUTPUT -
    "           THE  FULL BACKUP OF ''DISK' IS COMPLETE"
$  DISMOUNT/NOUNLOAD MUA0:
$!
$ ELSE
$!
$!  Do incremental backup
$!
$  ALLOCATE MUA0:
$  INITIALIZE MUA0: INCRBK
$  BACKUP /REWIND /RECORD /IGNORE=INTERLOCK -
  /MODIFIED /SINCE=BACKUP 'DISK'[*...]*.*;* MUA0:INCRBK.BCK
$  WRITE SYS$OUTPUT -
    "            THE INCREMENTAL BACKUP OF ''DISK' IS COMPLETE"
$  DISMOUNT/NOUNLOAD MUA0:
$ ENDIF
$!
$ SUBMIT /AFTER=TOMORROW SYS$MANAGER:BACKUP.COM
$!
$ EXIT
```

Test

QUESTIONS

Write the letter of the best answer in the space next to each of the following questions.

1. _____Suppose you create a user directory 1DIA200:[NEILSEN] and forget to specify that user NEILSEN should own it. What command would you use to correct the ownership of the directory?

   a. SET FILE /OWNER=NEILSEN 1DIA200:[NEILSEN]

   b. SET FILE /OWNER=NEILSEN 1DIA200:[000000]NEILSEN.DIR

   c. SET FILE /BY_OWNER=NEILSEN 1DIA200:[000000]NEILSEN.DIR

   d. SET FILE /BY_OWNER=NEILSEN 1DIA200:NEILSEN

2. _____Which BACKUP command qualifier is used to list the contents of a save set on tape?

   a. /LIST

   b. /SELECT

   c. /SINCE

   d. /SAVE_SET

3. _____What command procedure automates the procedure for registering or amending a license?

   a. SYS$UPDATE:VMSINSTAL.COM

   b. SYS$UPDATE:VMSLICENSE.COM

   c. SYS$SYSTEM:AUTHORIZE.COM

   d. SYS$UPDATE:AUTOGEN.COM

4. _____When do you usually need to install a product's license key?

   a. After you install the product software

   b. Before you install the product software

   c. Before you read the product documentation

   d. After users begin to use the product

5. _____A user mistypes his password five times. Thereafter, because of break-in detection, he cannot log in even if he types the correct password. What command can help you enable him to log in again?

   a. DELETE /INTRUSION_RECORD

   b. SET TERMINAL /SECURE_SERVER

   c. SET PASSWORD /SYSTEM

   d. MONITOR /SYSTEM

6. _____You create an image backup of a disk on Wednesday, on a tape labeled WED. You make incremental backups on Thursday on a tape labeled THU, on Friday on a tape labeled FRI, and on Monday on a tape labeled MON.

   On Tuesday afternoon the disk fails. After it is replaced, you want to restore it to the most recent possible state; in what order should you restore the backup tapes?

   a. THU, then FRI, then MON, then WED

   b. MON, then FRI, then THU, then WED

   c. WED, then THU, then FRI, then MON

   d. WED, then MON, then FRI, then THU

7. _____What BACKUP qualifiers are necessary to achieve incremental backups?

   a. /IMAGE, /INCREMENTAL

   b. /REWIND, /SINCE

   c. /INITIALIZE, /RECORD

   d. /RECORD, /SINCE

8. _____A user complains that an application does not work, and the error message is "RMS-E-FLK, file currently locked by another user" (in other words, the file is open and cannot be opened by two processes at once). What command would help you find out what other process has the file locked?

   a. SHOW DEVICE /FULL device-name

   b. DIRECTORY /FULL file-spec

   c. SHOW DEVICE /FILES device-name

   d. DIRECTORY /OPEN file-spec

9. _____ Using the Authorize utility, which flag would you set to restrict an account to a specific login command procedure?

   a. CAPTIVE

   b. DEFCLI

   c. DISCTLY

   d. LOCKPWD

10. _____What command displays all the logical names in the group table for your UIC group?

   a. SHOW LOGICAL /UIC

   b. SHOW LOGICAL /TABLE=GROUP

   c. SHOW LOGICAL /GROUP

   d. SHOW LOGICAL /SYSTEM

11. Several commands and qualifiers are available for monitoring print queues and jobs. For each of the functions below, select the command and qualifier that best implements that function and write its letter in the space provided. Each selection can be used once, more than once, or not at all.

	Function	Command/Qualifier
____	List all the queues on the system	a. SHOW ENTRY
____	List the print jobs that belong to a particular user	b. SHOW ENTRY /FULL entry-number
____	List the print and batch jobs that belong to you	c. SHOW ENTRY /DEVICE /USER_NAME=user-name
____	List all the entries in all the queues on the system	d. SHOW QUEUE
____	Find out the job limit on a particular queue	e. SHOW QUEUE /FULL queue-name
____	Find out what file is being printed in a particular entry	f. SHOW QUEUE /ALL
____	Find out what time a job was submitted for printing	

12. Many DCL features are useful in writing command procedures. For each of the functions below, select the command or qualifier that best implements that function and write its letter in the space provided. Each selection can be used once, more than once, or not at all.

	Function		Command/Qualifier
____	Putting the results of the START/QUEUE command in a file	a.	DEFINE
____	Putting the results of a DIRECTORY command in a file	b.	/OUTPUT=filename
____	Displaying the content of a symbol at a terminal	c.	TYPE symbol
____	Sending a copy of a file to the SYSTEM account	d.	MAIL
____	Putting the results of the ACCOUNTING command into a file	e.	WRITE symboL
____	Asking the user whether he or she wants to see some information (Y/N)	f.	INQUIRE
____	Displaying information that has been obtained from the user		

13. _____ If you wanted to know the amount of free space and total space on a disk volume, which command would you use?

a. SHOW VOLUME

b. SHOW QUOTA

c. SHOW DEVICE/FULL

d. SHOW ALL

14. _____ As a system manager, you would use disk quota rebuild for what purpose?

a. To reconstruct the usage counts for all entries on the volume

b. To reenable disk quotas on the volume after system failure

c. To rebuild the disk quota file after the disk volume was improperly dismounted

15. _____A process that controls individual printers on a VMS system is called:

a. A job controller

b. A print symbiont

c. A cluster server

d. An ancillary control process (ACP)

16. _____ What SYSMAN command is used to create a disk quota file on a volume?

a. DISKQUOTA CREATE

b. DISKQUOTA REBUILD

c. CREATE DISKQUOTA

d. VOLUME CREATE

17. _____Which of the following NCP commands manipulates the volatile database?

a. PURGE

b. DEFINE

c. LIST

d. CLEAR

18. _____ Which of the following commands creates a new entry in the permanent network database?

a. SET NODE ROGER ADDRESS 2.4

b. SET NODE 2.4 NAME ROGER

c. DEFINE NODE 2.4 NAME ROGER

d. DEFINE ADDRESS 2.4 NAME ROGER

19. _____ If you want to ask node MASTER what the address of node FIRST is, what NCP command could you use?

  a. SET HOST MASTER

  b. TELL MASTER SHOW NODE FIRST

  c. TELL MASTER SET EXECUTOR FIRST

  d. SET EXECUTOR NODE FIRST

20. _____Suppose you perform the following actions:

  1. Boot the cluster nodes BARNUM and RNGLNG, which are connected to the CI bus.

  2. Install a new disk drive, named 1DUA33:, on the HSC unit.

  3. Initialize 1DUA33:.

  4. Enter the MOUNT/CLUSTER command on BARNUM to mount 1DUA33:.

  5. Boot the node BAILEY, which is also connected to the CI bus.

  6. Boot the node HORSE, which is a satellite.

  Which nodes have 1DUA33: mounted after this sequence of actions?

  a. BARNUM

  b. BARNUM, RNGLNG

  c. BARNUM, RNGLNG, BAILEY

  d. BARNUM, RNGLNG, BAILEY, HORSE

21. _____In a command procedure, parameters P1 through P8 are available for use as:

  a. Logical names

  b. DCL symbols

  c. Character strings

  d. File specifications

22. _____In a command procedure, you can obtain the current system time by means of:

a. A predefined symbol

b. A logical name

c. A lexical function

d. A parameter

23. On the line next to each expression below, write the value (0 or 1) of the expression. Assume that:

- DAY is a symbol containing the string "Tuesday"
- TUESDAY is a symbol containing the integer 3
- VALUE is a symbol containing the integer 53

a. _____ DAY .EQS. "TUESDAY"

b. _____ DAY .EQS. TUESDAY

c. _____ VALUE .GT. TUESDAY

d. _____ 53 .EQ. VALUE

e. _____ DAY .EQS. "Tuesday"

f. _____ "THURSDAY" .GTS. "TUESDAY"

g. _____ VALUE .LE. 0

h. _____ TUESDAY .GT. 7

24. _____If the symbol VALUE contains the integer 5 on entry to the following DCL code, what value does TOTAL have after the code has executed?

```
$ IF VALUE .GT. 2
$ THEN
$     TOTAL = VALUE
$     TOTAL = TOTAL + 5
$ ELSE TOTAL = VALUE + 3
$ ENDIF
```

a. 4

b. 5

c. 6

d. 7

25. _____Which logical name table is available to all processes and subprocesses on a system?

a. LNM$PROCESS

b. LNM$JOB

c. LNM$GROUP

d. LNM$SYSTEM

26. _____Which of the following command statements will substitute a device's logical name (for example, GEORGE) for its physical name (for example, DUAO) in system message displays?

a. DEFINE GEORGE DUA0:

b. DEFINE/TRANSLATION_ATTRIBUTES=CONCEALED GEORGE DUA0:

c. DEFINE/TRANSLATION_ATTRIBUTES=TERMINAL GEORGE DUA0:

d. ASSIGN/TRANSLATION_ATTRIBUTES=TERMINAL GEORGE DUA0:

27. _____Which of the following would you use to cause the VMS system to record events such as process deletion, print job completion, login failure, and batch job completion?

a. SET ACCOUNTING

b. SET ACCOUNTING/RECORD

c. ACCOUNTING

d. ACCOUNTING/RECORD

28. _____Which of the following should be used to upgrade or update system layered product software?

a. SYS$UPDATE

b. VMSINSTAL

c. SYSGEN

d. INSTALL

29. _____Which of the following commands displays (on one screen) several of the most important classes of information a system manager can monitor?

   a. MONITOR PROCESSES /TOPCPU

   b. MONITOR STATES

   c. MONITOR SYSTEM

   d. MONITOR ALL_CLASSES

30. _____What DCL command most quickly shows you which nodes are currently members of the cluster?

   a. SHOW CLUSTER

   b. SHOW MEMBERS

   c. MONITOR NODES

   d. MONITOR CLUSTER

31. _____What DCL command most easily shows you what program a process is running?

   a. SHOW IMAGE

   b. SHOW USERS

   c. MONITOR PROCESSES

   d. SHOW PROCESS/CONTINUOUS

32. _____What DCL command do you use to find out whether the DECnet software is running?

   a. SHOW DECNET

   b. MONITOR CLUSTER

   c. SHOW NETWORK

   d. SHOW PROCESS

33. _____A disk device is connected to cluster node DD, which has allocation class 44. The device has unit number 33 and device type DU. Which of the following is a valid name for the device?

a. DDDUA44

b. 44DDA33

c. 33DUA44

d. 44DUA33

34. _____Which of the following control break-in detection?

a. Symbols

b. Identifiers

c. Logical names

d. System parameters

35. _____What file do you edit to make parameter changes permanently known to AUTOGEN?

a. SYS$MANAGER:VMSIMAGES.DAT

b. SYS$UPDATE:AUTOGEN.DAT

c. SYS$SYSTEM:SETPARAMS.DAT

d. SYS$SYSTEM:MODPARAMS.DAT

36. _____Which DCL command is used to create a queue?

a. INITIALIZE/QUEUE

b. CREATE/QUEUE

c. START/QUEUE

d. ASSIGN/QUEUE

37. _____In which file are terminal speeds, queues, and other system management related operations usually specified?

a. SYLOGIN.COM

b. SYSTARTUP_V5.COM

c. STARTUP.COM

d. SYLOGICALS.COM

38. _____Which procedure should mount site-specific volumes?

a. SYSTARTUP_V5.COM

b. STARTUP.COM

c. SYLOGICALS.COM

d. SYCONFIG.COM

39. _____Which procedure should define standard logical names such as SYS$LOGIN, SYS$ANNOUNCE, and SYS$WELCOME?

a. SYLOGICALS.COM

b. SYCONFIG.COM

c. SYSTARTUP_V5.COM

d. STARTUP.COM

40. _____In which table are shareable logical name tables cataloged?

a. LNM$SYSTEM_TABLE

b. LNM$GROUP

c. LNM$JOB

d. LNM$PROCESS_TABLE

41. _____Before an active queue can be deleted, it must first be:

- **a.** Initialized
- **b.** Stopped
- **c.** Spooled
- **d.** Started

ANSWERS

1. __b__ Suppose you create a user directory 1DIA200:[NEILSEN] and forget to specify that user NEILSEN should own it. What command would you use to correct the ownership of the directory?

   a. SET FILE /OWNER=NEILSEN 1DIA200:[NEILSEN]

   b. SET FILE /OWNER=NEILSEN 1DIA200:[000000]NEILSEN.DIR

   c. SET FILE /BY_OWNER=NEILSEN 1DIA200:[000000]NEILSEN.DIR

   d. SET FILE /BY_OWNER=NEILSEN 1DIA200:NEILSEN

2. __a__ Which BACKUP command qualifier is used to list the contents of a save set on tape?

   a. /LIST

   b. /SELECT

   c. /SINCE

   d. /SAVE_SET

3. __b__ What command procedure automates the procedure for registering or amending a license?

   a. SYS$UPDATE:VMSINSTAL.COM

   b. SYS$UPDATE:VMSLICENSE.COM

   c. SYS$SYSTEM:AUTHORIZE.COM

   d. SYS$UPDATE:AUTOGEN.COM

4. __b__ When do you usually need to install a product's license key?

   a. After you install the product software

   b. Before you install the product software

   c. Before you read the product documentation

   d. After users begin to use the product

5. __b__ A user mistypes his password five times. Thereafter, because of break-in detection, he cannot log in even if he types the correct password. What command can help you enable him to log in again?

   a. DELETE /INTRUSION_RECORD

   b. SET TERMINAL /SECURE_SERVER

   c. SET PASSWORD /SYSTEM

   d. MONITOR /SYSTEM

6. __d__ You create an image backup of a disk on Wednesday, on a tape labeled WED. You make incremental backups on Thursday on a tape labeled THU, on Friday on a tape labeled FRI, and on Monday on a tape labeled MON.

   On Tuesday afternoon the disk fails. After it is replaced, you want to restore it to the most recent possible state; in what order should you restore the backup tapes?

   a. THU, then FRI, then MON, then WED

   b. MON, then FRI, then THU, then WED

   c. WED, then THU, then FRI, then MON

   d. WED, then MON, then FRI, then THU

7. __d__ What BACKUP qualifiers are necessary to achieve incremental backups?

   a. /IMAGE, /INCREMENTAL

   b. /REWIND, /SINCE

   c. /INITIALIZE, /RECORD

   d. /RECORD, /SINCE

8. __c__ A user complains that an application does not work, and the error message is "RMS-E-FLK, file currently locked by another user" (in other words, the file is open and cannot be opened by two processes at once). What command would help you find out what other process has the file locked?

   a. SHOW DEVICE /FULL device-name

   b. DIRECTORY /FULL file-spec

   c. SHOW DEVICE /FILES device-name

   d. DIRECTORY /OPEN file-spec

9. __a__ Using the Authorize utility, which flag would you set to restrict an account to a specific login command procedure?

   a. CAPTIVE

   b. DEFCLI

   c. DISCTLY

   d. LOCKPWD

10. __c__ What command displays all the logical names in the group table for your UIC group?

   a. SHOW LOGICAL /UIC

   b. SHOW LOGICAL /TABLE=GROUP

   c. SHOW LOGICAL /GROUP

   d. SHOW LOGICAL /SYSTEM

11. Several commands and qualifiers are available for monitoring print queues and jobs. For each of the functions below, select the command and qualifier that best implements that function and write its letter in the space provided. Each selection can be used once, more than once, or not at all.

	Function	Command/Qualifier
__d__	List all the queues on the system	a. SHOW ENTRY
__c__	List the print jobs that belong to a particular user	b. SHOW ENTRY /FULL entry-number
__a__	List the print and batch jobs that belong to you	c. SHOW ENTRY /DEVICE /USER_NAME=user-name
__f__	List all the entries in all the queues on the system	d. SHOW QUEUE
__e__	Find out the job limit on a particular queue	e. SHOW QUEUE /FULL queue-name
__b__	Find out what file is being printed in a particular entry	f. SHOW QUEUE /ALL
__b__	Find out what time a job was submitted for printing	

12. Many DCL features are useful in writing command procedures. For each of the functions below, select the command or qualifier that best implements that function and write its letter in the space provided. Each selection can be used once, more than once, or not at all.

	Function	Command/Qualifier
__a__	Putting the results of the START/QUEUE command in a file	a. DEFINE
__b__	Putting the results of a DIRECTORY command in a file	b. /OUTPUT=filename
__e__	Displaying the content of a symbol at a terminal	c. TYPE symbol
__d__	Sending a copy of a file to the SYSTEM account	d. MAIL
__b__	Putting the results of the ACCOUNTING command into a file	e. WRITE symboL
__f__	Asking the user whether he or she wants to see some information (Y/N)	f. INQUIRE
__e__	Displaying information that has been obtained from the user	

13. __c__ If you wanted to know the amount of free space and total space on a disk volume, which command would you use?

a. SHOW VOLUME

b. SHOW QUOTA

c. SHOW DEVICE/FULL

d. SHOW ALL

14. __a__ As a system manager, you would use disk quota rebuild for what purpose?

a. To reconstruct the usage counts for all entries on the volume

b. To reenable disk quotas on the volume after system failure

c. To rebuild the disk quota file after the disk volume was improperly dismounted

15. __b__ A process that controls individual printers on a VMS system is called:

   a. A job controller

   b. A print symbiont

   c. A cluster server

   d. An ancillary control process (ACP)

16. __a__ What SYSMAN command is used to create a disk quota file on a volume?

   a. DISKQUOTA CREATE

   b. DISKQUOTA REBUILD

   c. CREATE DISKQUOTA

   d. VOLUME CREATE

17. __d__ Which of the following NCP commands manipulates the volatile database?

   a. PURGE

   b. DEFINE

   c. LIST

   d. CLEAR

18. __c__ Which of the following commands creates a new entry in the permanent network database?

   a. SET NODE ROGER ADDRESS 2.4

   b. SET NODE 2.4 NAME ROGER

   c. DEFINE NODE 2.4 NAME ROGER

   d. DEFINE ADDRESS 2.4 NAME ROGER

19. __b__ If you want to ask node MASTER what the address of node FIRST is, what NCP command could you use?

a. SET HOST MASTER

b. TELL MASTER SHOW NODE FIRST

c. TELL MASTER SET EXECUTOR FIRST

d. SET EXECUTOR NODE FIRST

20. __b__ Suppose you perform the following actions:

1. Boot the cluster nodes BARNUM and RNGLNG, which are connected to the CI bus.
2. Install a new disk drive, named 1DUA33:, on the HSC unit.
3. Initialize 1DUA33:.
4. Enter the MOUNT/CLUSTER command on BARNUM to mount 1DUA33:.
5. Boot the node BAILEY, which is also connected to the CI bus.
6. Boot the node HORSE, which is a satellite.

Which nodes have 1DUA33: mounted after this sequence of actions?

a. BARNUM

b. BARNUM, RNGLNG

c. BARNUM, RNGLNG, BAILEY

d. BARNUM, RNGLNG, BAILEY, HORSE

21. __b__ In a command procedure, parameters P1 through P8 are available for use as:

a. Logical names

b. DCL symbols

c. Character strings

d. File specifications

22. __c__ In a command procedure, you can obtain the current system time by means of:

a. A predefined symbol

b. A logical name

c. A lexical function

d. A parameter

23. On the line next to each expression below, write the value (0 or 1) of the expression. Assume that:

- DAY is a symbol containing the string "Tuesday"
- TUESDAY is a symbol containing the integer 3
- VALUE is a symbol containing the integer 53

a. __0__ DAY .EQS. "TUESDAY"

b. __0__ DAY .EQS. TUESDAY

c. __1__ VALUE .GT. TUESDAY

d. __1__ 53 .EQ. VALUE

e. __1__ DAY .EQS. "Tuesday"

f. __0__ "THURSDAY" .GTS. "TUESDAY"

g. __0__ VALUE .LE. 0

h. __1__ TUESDAY .GT. 7

24. __c__ If the symbol VALUE contains the integer 5 on entry to the following DCL code, what value does TOTAL have after the code has executed?

```
$ IF VALUE .GT. 2
$ THEN
$     TOTAL = VALUE
$     TOTAL = TOTAL + 5
$ ELSE TOTAL = VALUE + 3
$ ENDIF
```

a. 4

b. 5

c. 6

d. 7

25. __d__ Which logical name table is available to all processes and subprocesses on a system?

a. LNM$PROCESS

b. LNM$JOB

c. LNM$GROUP

d. LNM$SYSTEM

26. __b__ Which of the following command statements will substitute a device's logical name (for example, GEORGE) for its physical name (for example, DUAO) in system message displays?

a. DEFINE GEORGE DUA0:

b. DEFINE/TRANSLATION_ATTRIBUTES=CONCEALED GEORGE DUA0:

c. DEFINE/TRANSLATION_ATTRIBUTES=TERMINAL GEORGE DUA0:

d. ASSIGN/TRANSLATION_ATTRIBUTES=TERMINAL GEORGE DUA0:

27. __a__ Which of the following would you use to cause the VMS system to record events such as process deletion, print job completion, login failure, and batch job completion?

a. SET ACCOUNTING

b. SET ACCOUNTING/RECORD

c. ACCOUNTING

d. ACCOUNTING/RECORD

28. __b__ Which of the following should be used to upgrade or update system layered product software?

a. SYS$UPDATE

b. VMSINSTAL

c. SYSGEN

d. INSTALL

29. ___c___ Which of the following commands displays (on one screen) several of the most important classes of information a system manager can monitor?

a. MONITOR PROCESSES /TOPCPU

b. MONITOR STATES

c. MONITOR SYSTEM

d. MONITOR ALL_CLASSES

30. ___a___ What DCL command most quickly shows you which nodes are currently members of the cluster?

a. SHOW CLUSTER

b. SHOW MEMBERS

c. MONITOR NODES

d. MONITOR CLUSTER

31. ___d___ What DCL command most easily shows you what program a process is running?

a. SHOW IMAGE

b. SHOW USERS

c. MONITOR PROCESSES

d. SHOW PROCESS/CONTINUOUS

32. ___c___ What DCL command do you use to find out whether the DECnet software is running?

a. SHOW DECNET

b. MONITOR CLUSTER

c. SHOW NETWORK

d. SHOW PROCESS

33. __d__ A disk device is connected to cluster node DD, which has allocation class 44. The device has unit number 33 and device type DU. Which of the following is a valid name for the device?

   a. DDDUA44

   b. 44DDA33

   c. 33DUA44

   d. 44DUA33

34. __d__ Which of the following control break-in detection?

   a. Symbols

   b. Identifiers

   c. Logical names

   d. System parameters

35. __d__ What file do you edit to make parameter changes permanently known to AUTOGEN?

   a. SYS$MANAGER:VMSIMAGES.DAT

   b. SYS$UPDATE:AUTOGEN.DAT

   c. SYS$SYSTEM:SETPARAMS.DAT

   d. SYS$SYSTEM:MODPARAMS.DAT

36. __a__ Which DCL command is used to create a queue?

   a. INITIALIZE/QUEUE

   b. CREATE/QUEUE

   c. START/QUEUE

   d. ASSIGN/QUEUE

37. __b__ In which file are terminal speeds, queues, and other system management related operations usually specified?

   a. SYLOGIN.COM

   b. SYSTARTUP_V5.COM

   c. STARTUP.COM

   d. SYLOGICALS.COM

38. __a__ Which procedure should mount site-specific volumes?

   a. SYSTARTUP_V5.COM

   b. STARTUP.COM

   c. SYLOGICALS.COM

   d. SYCONFIG.COM

39. __a__ Which procedure should define standard logical names such as SYS$LOGIN, SYS$ANNOUNCE, and SYS$WELCOME?

   a. SYLOGICALS.COM

   b. SYCONFIG.COM

   c. SYSTARTUP_V5.COM

   d. STARTUP.COM

40. __b__ In which table are shareable logical name tables cataloged?

   a. LNM$SYSTEM_TABLE

   b. LNM$GROUP

   c. LNM$JOB

   d. LNM$PROCESS_TABLE

41. __b__ Before an active queue can be deleted, it must first be:

a. Initialized

b. Stopped

c. Spooled

d. Started

INDEX

J

L

Q

R

S

V

W